THE CAPACITY LIMITS OF SEQUENTIAL KNOWLEDGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential knowledge editing updates a language model one fact at a time without retraining it from scratch. Most recent methods make each edit by changing the same feed-forward weights. This works well for shorter edit sequences, but performance can fall sharply as more updates accumulate. We study a simple alternative. Instead of writing every edit into the model weights, we keep the model frozen and store the optimized hidden value for each edit in an external memory. We call the resulting method MoKE. Several context-dependent keys are stored for each edit so that paraphrases can retrieve the same value. On the standard 3,000-edit protocol with GPT2-XL and Llama3-8B, MoKE gives the best reported result on 15 of 18 metrics. In a controlled comparison with DeltaEdit using the same code, machine, precision, seed, and edit stream, MoKE improves CounterFact specificity by 35.0 and 49.8 points. At 10,000 edits, MoKE retains 97.8 efficacy while state-of-the-art falls to 13.1, and this endpoint is replicated with a second seed. Keeping the base model frozen also preserves downstream accuracy much better after editing. External memory has clear costs. Memory grows linearly with the number of stored keys, the retrieval threshold must be calibrated, and specificity still falls as the memory grows. Direct measurements show that this loss is not explained by an increasing rate of paraphrases being matched to another edit. These results show that the scale of sequential editing depends strongly on where the edits are stored. Code: `https://anonymous.4open.science/r/moke-editing-84FD/README.md`

1 INTRODUCTION

Large language models contain a large amount of factual knowledge in their parameters (Petroni et al., 2019; Roberts et al., 2020). Some of that knowledge becomes outdated. Knowledge editing aims to change a fact without retraining the whole model (De Cao et al., 2021; Meng et al., 2022; 2023; Yao et al., 2023). The difficult case is not one correction but a long sequence of them. Each new edit must work without undoing earlier edits or damaging the model on unrelated inputs (Gupta et al., 2024b; Yang et al., 2024; Gu et al., 2024).

Most recent work has focused on making repeated weight updates safer (Meng et al., 2022; 2023; Ma et al., 2025; Fang et al., 2025; Cai & Cao, 2024; Cao et al., 2026; Liu et al., 2026; Dai et al., 2026; Yang et al., 2026). These methods differ in how they compute an update, but they share one basic choice. The new fact is stored by changing model parameters.

We study that choice directly. Locate-then-edit methods first optimize a hidden value that makes the model produce the requested answer. They then solve for a weight update that makes the model produce that value. We keep the same optimized value and the same edit layer, but change what happens next. DeltaEdit writes the edit into the model weights. MoKE keeps the model weights frozen and stores the optimized value in an external memory that is consulted at inference. This gives a controlled comparison between two ways of keeping the same edit.

Figure 1 shows the difference as the sequence grows on GPT2-XL and CounterFact. DeltaEdit Cao et al. (2026) is still strong at 2,500 edits but reaches 13.13 efficacy at 10,000. MoKE ends at 97.81 on the first seed and 97.69 on the second. Its specificity does decline as memory grows. The two storage choices therefore show very different scaling under the same editing procedure.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

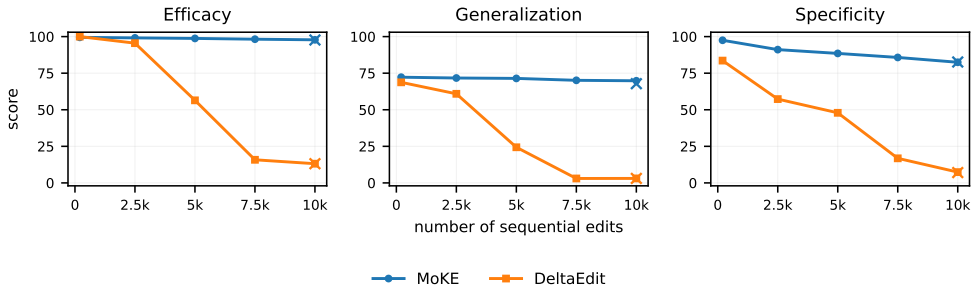

Figure 1: Sequential editing as the number of edits grows on GPT2-XL and CounterFact. Lines show the first seed. Crosses at 10,000 edits show a second run. DeltaEdit loses most of its editing performance between 2,500 and 7,500 edits. MoKE keeps efficacy high while generalization and specificity decline gradually.

External memory is not always better. On Llama3-8B and ZsRE, DeltaEdit Cao et al. (2026) already reaches 95.50 generalization after 3,000 edits in our controlled run. MoKE reaches 90.80. This negative result is important. Moving edits outside the weights is most useful when repeated parameter updates are already causing substantial interference. When weight editing remains stable, external retrieval can introduce an unnecessary cost.

We call the resulting method **MoKE**. MoKE keeps the model frozen, stores the optimized target value for each edit, and uses keys from several contexts to improve paraphrase retrieval. On CounterFact, these additional context keys improve generalization by 13.02 points on GPT2-XL with no measurable specificity loss at the reported precision. The model weights remain bit-identical after editing, which gives a direct explanation for the strong preservation of downstream behavior.

Our main findings are threefold.

1. We compare repeated weight editing with external memory while keeping the target-value optimization and edit layer fixed. This isolates where the edit is stored rather than changing the whole editing pipeline.
2. MoKE is strong at the standard 3,000-edit scale and behaves very differently beyond it. It gives the best reported result on 15 of 18 metrics and retains about 97.8 efficacy at 10,000 edits, where DeltaEdit Cao et al. (2026) falls to 13.1.
3. We measure the costs and failure cases of external memory. Memory grows linearly, threshold choice matters, Llama3-8B on ZsRE is a real loss, and direct retrieval measurements do not support the simple explanation that specificity falls because paraphrases are increasingly matched to another edit.

2 Related Work

Methods that update model weights. ROME, MEMIT, and related methods directly modify a small set of model parameters (De Cao et al., 2021; Mitchell et al., 2022a; Meng et al., 2022; 2023). Applied sequentially, such updates can interfere with earlier edits and reduce general model capabilities (Gupta et al., 2024b;a; Yang et al., 2024; Gu et al., 2024; Li et al., 2024). Later work reduces this damage through constrained perturbations, layer selection, null-space projection, update orthogonalization, weight geometry, and improved target alignment (Ma et al., 2025; Hu et al., 2024; Jiang et al., 2025; Fang et al., 2025; Cai & Cao, 2024; Cao et al., 2026; Liu et al., 2026; Dai et al., 2026; Yang et al., 2026). We instead keep the optimized target representation fixed and ask whether it should be stored in the weights at all.

Methods that keep editing state outside the base model. A second group of methods uses additional state at inference rather than permanently changing the original parameters. SERAC learns a scope classifier and a counterfactual model (Mitchell et al., 2022b). GRACE stores key-value entries at an internal layer (Hartvigsen et al., 2023). Transformer-Patcher adds neurons (Huang et al., 2023).

MELO retrieves LoRA blocks from a vector index (Yu et al., 2024). WISE adds a separate memory and routes queries between memories (Wang et al., 2024). Other work uses in-context editing or retrieval-augmented prompts (Zheng et al., 2023; Chen et al., 2024). These methods are usually evaluated under protocols that do not match the 3,000-edit GPT2-XL and Llama3-8B setup used by recent sequential weight-editing work. We therefore do not report an official rerun when there is no overlapping published result that can be used to check the implementation. Appendix G explains this choice and compares MoKE with a single-key memory inside our own implementation.

Retrieval and external memory. Nearest-neighbor language models and retrieval-augmented generation also keep knowledge outside the model parameters (Khandelwal et al., 2020; Lewis et al., 2020). MoKE differs in what is retrieved. It does not retrieve text. It retrieves the optimized hidden value used by locate-then-edit methods and inserts that value at the same internal layer. This shared target representation lets us compare weight editing with external memory without changing the definition of the edit.

3 Where should a sequential edit be stored?

We write an edit as $\mathcal{E}_j = (s_j, r_j, o_j, o_j^*)$. It asks the model to answer a prompt $p(s_j, r_j)$ with a new object o_j^* instead of o_j. Locate-then-edit methods first find a hidden value that would make the requested answer likely. Following ROME, MEMIT, AlphaEdit and DeltaEdit, we optimize

$$z_j^* = \arg\min_z \ \mathbb{E}_{c\sim\mathcal{T}} \left[-\log P_{\mathcal{M}(h^\ell:=z)}\big(o_j^* \mid c \oplus p(s_j, r_j)\big)\right]. \tag{1}$$

The practical objective uses the same regularization terms and optimization budget as the reference editor. Equation 1 specifies the hidden value that should produce the new answer. It does not require that this value be stored by changing model weights.

3.1 Repeated weight updates accumulate

DeltaEdit and related methods solve an update Δ_j to a feed-forward output matrix. For a rank-one update $\Delta_j = \alpha_j \beta_j^\top$, the matrix after T edits is

$$W_T = W_0 + \sum_{j=1}^{T} \Delta_j. \tag{2}$$

For a probe k_e associated with edit e, the edited layer returns

$$W_T k_e = W_0 k_e + \Delta_e k_e + \sum_{j\neq e} (k_e^\top \beta_j)\alpha_j. \tag{3}$$

The final term contains contributions from every other edit. Recent sequential editors try to make these contributions smaller or less harmful (Fang et al., 2025; Cai & Cao, 2024; Cao et al., 2026; Liu et al., 2026). The underlying structure remains the same. Every new edit changes parameters that already contain earlier edits, and the number of accumulated updates grows with the edit sequence.

3.2 External memory avoids repeated weight changes

The alternative is to leave W_0 unchanged. Let $\mathcal{C} = \{(k_i, v_i)\}_{i=1}^M$ contain keys and stored hidden values at layer ℓ. For each token hidden state h_t, we find the most similar key by cosine similarity. If that similarity is above a threshold τ, the corresponding stored value replaces the current hidden state. Otherwise the model continues normally.

$$h_t \leftarrow \begin{cases} v_{i^*}, & i^* = \arg\max_i \cos(h_t, k_i), \quad \cos(h_t, k_{i^*}) > \tau, \\ h_t, & \text{otherwise.} \end{cases} \tag{4}$$

This has two useful properties. A retrieved query receives one stored value rather than a sum of all previous edit updates. Also, if no stored edit is retrieved, the computation is exactly the same as in the unedited model. Section 5.4 checks the second property directly by verifying that the model weights remain bit-identical.

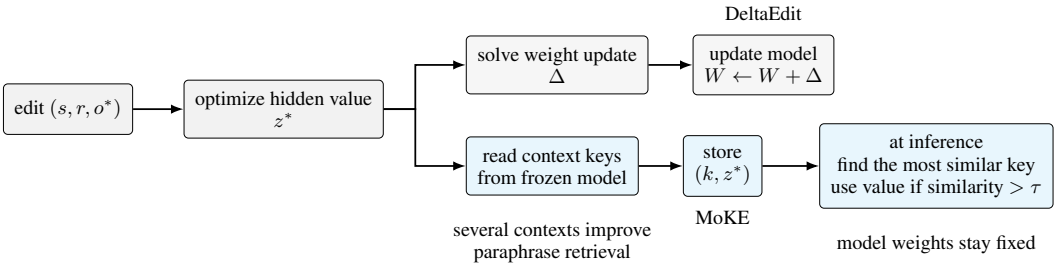

Figure 2: MoKE changes how the optimized edit is stored. DeltaEdit solves a weight update and adds it to the model. MoKE keeps the model frozen, stores the same optimized hidden value in external memory, and retrieves that value when a query is sufficiently similar to one of the stored keys.

External memory has its own limits. The memory grows with the number of edits. If τ is too low, unrelated prompts can retrieve an edit. If it is too high, paraphrases can fail to retrieve the intended edit. Adding more context keys can improve paraphrase generalization, but it also increases memory and retrieval cost.

One obvious concern is that a larger memory might increasingly send paraphrases to a key belonging to a different edit. We measure this directly. Across a 15-fold increase in memory size, the fraction of paraphrase queries assigned to a key from another edit changes only from 44.80 to 45.43 percent, while specificity falls from 97.50 to 93.29. This measurement does not explain the loss in specificity. We therefore make no claim that retrieval confusion is the mechanism behind the observed degradation. The experiments establish that external memory also has a capacity limit, but the cause of its remaining specificity loss is still open.

4 MoKE

MoKE makes one main change to the locate-then-edit pipeline. Both DeltaEdit and MoKE optimize the same z_j^* in Equation 1. DeltaEdit then solves a parameter update. MoKE stores z_j^* in external memory and leaves the model parameters unchanged.

What is stored. For edit j, MoKE stores $v_j = z_j^*$. This optimized value is essential. On GPT2-XL and CounterFact, reducing the target optimization from 20 steps to 5 lowers efficacy from 98.93 to 41.57. One step gives 0.13 efficacy. MoKE therefore depends on the same target-value optimization that makes locate-then-edit methods effective.

How an edit is indexed. The basic key is the hidden state at the last subject token

$$k_j = h^\ell\big(p(s_j, r_j)\big)\big|_{\text{last token of } s_j}. \tag{5}$$

Because MoKE never changes the model weights, every key is computed in the same representation space. A key stored early in the sequence is not changed indirectly by later edits.

Keys from several contexts. One key from the original prompt gives limited coverage of paraphrases. We therefore use the prefix contexts $\mathcal{T}$ that are already used when optimizing z_j^*. Each context gives another key for the same stored value

$$\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ \left(h^\ell(c \oplus p(s_j, r_j))|_{s_j}, z_j^* \right) : c \in \mathcal{T} \right\}. \tag{6}$$

The default CounterFact setting uses seven keys per edit. At 3,000 edits this gives 21,000 keys. We refer to these as manifold keys in the method name, but use the simpler term context keys throughout the paper. Table 4 shows that they substantially improve paraphrase generalization on CounterFact and have almost no effect on ZsRE.

Retrieval threshold. At inference, MoKE applies Equation 4 at the same layer used for target-value optimization. The only new hyperparameter is τ. We also test a held-out procedure for selecting it. We reserve 600 edits for calibration, choose the threshold that maximizes the difference between paraphrase retrieval and unrelated-prompt retrieval, and then freeze the threshold for evaluation. On GPT2-XL and CounterFact this procedure chooses $\tau = 0.875$. Choosing the threshold directly on the evaluation split gives the same value under this criterion. Full results are in Appendix B.

Cost. Retrieval requires one similarity matrix product against the stored keys at the edited layer. With 21,000 keys, the measured decoding overhead is 0.3 percent on GPT2-XL and 3.5 percent on Llama3-8B when the normalized memory remains on the same device as the model. Memory use is 269 MB and 688 MB respectively and grows linearly with the number of keys.

5 EXPERIMENTS

5.1 SETUP

We follow the 3,000-edit sequential protocol used by AlphaEdit and DeltaEdit. Edits arrive one at a time on GPT2-XL (Radford et al., 2019) and Llama3-8B (Grattafiori et al., 2024). We evaluate CounterFact (Meng et al., 2022) and ZsRE (Levy et al., 2017). We keep the critical layers and target-value optimization from the reference configuration. The main change is only how the optimized value is stored. Along with the code provided, implementation details can be found in the appendix.

The standard metrics are efficacy, generalization, and specificity. Efficacy evaluates the edited prompt. Generalization evaluates paraphrases. Specificity evaluates unrelated neighborhood prompts and checks whether the original answer remains preferred. We report the same top-token and larger-probability variants used by the reference table. All scores are multiplied by 100.

Table 1 reports the published comparison table together with our MoKE results. The baseline rows are published values. MoKE is our run. Table 2 gives the controlled comparison with DeltaEdit. Both methods use the same codebase, machine, numerical precision, seed, edit stream, target-value optimization, and edit layers. The only difference is whether the optimized value is written into the model weights or stored in external memory.

5.2 RESULTS AFTER 3,000 EDITS

MoKE gives the best reported result on 15 of the 18 metrics in Table 1. The largest gains are on CounterFact specificity. GPT2-XL reaches 90.11 compared with the previous best of 53.28. Llama3-8B reaches 83.98 compared with 34.20. Efficacy remains close to 99 in both cases. The result is not uniformly positive. On GPT2-XL and ZsRE, MoKE specificity is 0.18 below the best published value. On Llama3-8B and ZsRE, the three-seed mean is lower in both generalization and specificity. The generalization gap is 0.72 points. With a measured standard deviation of 0.171 for that setting, this is a real loss rather than a statistical tie.

The controlled comparison in Table 2 shows what changes when the storage method changes. On CounterFact, MoKE improves specificity over DeltaEdit by 35.03 points for GPT2-XL and 49.77 points for Llama3-8B. GPT2-XL also gains 14.43 points in generalization. The Llama3-8B generalization scores are nearly identical. ZsRE gives the opposite result on Llama3-8B. DeltaEdit already reaches 98.97 efficacy and 95.50 generalization, while MoKE reaches 98.87 and 90.80. When repeated weight updates remain stable, external retrieval can reduce performance rather than improve it.

5.3 WHAT HAPPENS AT 10,000 EDITS

The 3,000-edit checkpoint hides the later failure of DeltaEdit on GPT2-XL and CounterFact. Figure 1 extends the same experiment to 10,000 edits. DeltaEdit efficacy is still 95.52 at 2,500 edits. It falls to 56.44 at 5,000, 15.80 at 7,500, and 13.13 at 10,000. Generalization falls to 3.07 and specificity to about 7.

Table 1: Sequential editing after 3,000 edits. Baseline rows are the published values from the DeltaEdit comparison table. MoKE uses our runs. Efficacy, generalization and specificity are shown as percentages. Bold marks the best value in each column, underlining marks the second best, and the subscript gives the improvement of the best result over the second best. The Llama3-8B ZsRE MoKE row is the mean of three seeds. The controlled comparison using the same experimental setup is in Table 2.

	CounterFact top			CounterFact larger			ZsRE		
Method	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.
GPT2-XL									
FT	27.17	8.30	2.22	69.93	50.43	18.24	5.13	4.43	0.28
ROME	0.53	0.38	0.41	53.93	49.55	84.56	44.81	41.12	7.52
MEMIT	25.73	16.85	12.71	67.80	60.85	68.99	26.80	25.38	13.51
PRUNE	7.93	6.45	8.91	60.00	56.13	80.94	2.94	2.77	3.49
RECT	56.77	31.13	18.11	88.30	74.80	72.85	39.50	36.13	15.50
AlphaEdit	85.93	46.23	53.18	98.37	88.05	92.73	93.19	84.97	23.55
DeltaEdit	93.80	54.37	53.28	98.97	91.10	92.91	95.26	88.48	**24.96**$_{+0.18}$
MoKE	**98.93**$_{+5.13}$	**71.28**$_{+16.91}$	**90.11**$_{+36.83}$	**99.57**$_{+0.60}$	**96.60**$_{+5.50}$	**98.21**$_{+5.30}$	**98.96**$_{+3.70}$	**93.22**$_{+4.74}$	24.78
Llama3-8B									
FT	9.60	3.63	0.23	81.00	66.93	12.20	6.45	6.01	24.31
ROME	0.00	0.00	0.00	63.37	59.50	24.77	2.95	2.82	1.22
MEMIT	0.00	0.00	0.00	50.33	50.37	81.62	0.00	0.00	0.00
PRUNE	0.00	0.00	0.00	51.20	51.20	17.14	0.00	0.00	0.00
RECT	0.00	0.00	0.00	49.23	49.23	16.07	0.00	0.00	0.00
AlphaEdit	77.07	53.92	17.59	93.83	84.75	64.44	94.49	**91.52**$_{+0.33}$	29.77
DeltaEdit	93.87	56.52	34.20	98.63	83.20	78.67	94.94	91.19	**30.08**$_{+0.31}$
MoKE	**99.03**$_{+5.16}$	**63.05**$_{+6.53}$	**83.98**$_{+49.78}$	**99.57**$_{+0.94}$	**88.77**$_{+4.02}$	**97.28**$_{+15.66}$	**98.77**$_{+3.83}$	90.80	28.87

Table 2: Controlled comparison between DeltaEdit and MoKE. Both methods use the same codebase, machine, precision, seed, edit stream, target-value optimization, and edit layers. The only difference is whether the optimized value is written into the model weights or stored in external memory. Bold marks the best value in each column, underlining marks the second best, and the subscript gives the improvement of the best result over the second best.

Model	Method and data	Eff.	Gen.	Spe.
GPT2-XL	DeltaEdit, CF	93.83	56.85	55.08
	MoKE, CF	**98.93**$_{+5.10}$	**71.28**$_{+14.43}$	**90.11**$_{+35.03}$
GPT2-XL	DeltaEdit, ZsRE	93.47	85.48	24.74
	MoKE, ZsRE	**98.96**$_{+5.49}$	**93.22**$_{+7.74}$	**24.78**$_{+0.04}$
Llama3-8B	DeltaEdit, CF	93.57	**63.32**$_{+0.27}$	34.21
	MoKE, CF	**99.03**$_{+5.46}$	63.05	**83.98**$_{+49.77}$
Llama3-8B	DeltaEdit, ZsRE	**98.97**$_{+0.10}$	**95.50**$_{+4.70}$	**30.50**$_{+1.63}$
	MoKE, ZsRE	98.87	90.80	28.87

MoKE also loses performance as more edits are stored, but the decline is much smaller. Specificity falls from 97.50 to 82.45 and generalization from 72.25 to 69.81, while efficacy remains 97.81. A second run ends at 97.69 efficacy, 67.82 generalization, and 82.65 specificity. The DeltaEdit endpoint repeats at 13.13 efficacy and 3.07 generalization. DeltaEdit is not retuned for 10,000 edits, so this is a scaling comparison under the adopted configuration rather than a universal ranking of long-sequence methods (Yang et al., 2026; Gu et al., 2026).

5.4 Keeping the weights fixed preserves general capabilities

Knowledge editing is useful only if the rest of the model continues to work. We evaluate five GLUE tasks after 3,000 CounterFact edits (Wang et al., 2019; Socher et al., 2013; Warstadt et al., 2019; Dolan & Brockett, 2005; Bentivogli et al., 2009; Williams et al., 2018). Table 3 uses 200 examples per task and reports zero-shot accuracy. MoKE matches the unedited Llama3-8B checkpoint to three decimals on four of five tasks and GPT2-XL on three of five. The remaining differences are at

Table 3: General capabilities after 3,000 CounterFact edits. Zero-shot accuracy on 200 examples per GLUE task. Bold marks the best value and underlining marks the second best.

Model	Method	SST-2	CoLA	MRPC	RTE	MNLI
GPT2-XL	unedited	**.535**	.650	.540	**.540**	.310
	MoKE	**.535**	.650	**.545**	**.540**	.305
	DeltaEdit	.515	**.665**	.350	.535	**.315**
Llama3-8B	unedited	**.845**	.665	**.685**	**.540**	**.375**
	MoKE	**.845**	.665	**.685**	**.540**	.365
	DeltaEdit	.735	**.670**	.680	**.540**	.315

Table 5: Component checks on GPT2-XL and CounterFact at 3,000 edits. All other settings are fixed. Bold marks the best value and underlining marks the second best within each component.

Question	Setting	Eff.	Gen.	Spe.
Target optimization	20 steps	**98.93**	**71.28**	90.11
	5 steps	41.57	24.57	96.63
	1 step	0.13	0.35	**99.71**
Retrieval layer	17	99.03	**71.10**	**93.29**
	15	**99.27**	70.28	92.79
	13	99.03	68.68	91.63
Key form	raw hidden	98.93	**71.28**	90.11
	mean centered	**99.17**	59.02	**92.34**

Table 4: Multiple context keys improve paraphrase generalization on CounterFact. The threshold and all other settings are fixed. The effect is a null on ZsRE. Bold marks the best value and underlining marks the second best within each setting.

Setting	Single key Gen.	Context keys Gen.	Δ Gen.	Δ Spe.
GPT2-XL / CF	58.27	**71.28**	+13.02	-0.02
Llama3-8B / CF	56.95	**62.25**	+5.30	-0.26
Llama3-8B / ZsRE	82.58	**82.76**	+0.18	+0.03

most two examples. Under the same evaluation, DeltaEdit loses 11.0 accuracy points on Llama3-8B SST-2 and 19.0 points on GPT2-XL MRPC.

We also check the weights directly. After 20 MoKE edits, the maximum absolute change in every edited feed-forward weight matrix is exactly zero. As a positive control, the same test detects changes from 2.26×10^{-2} to 1.08×10^{-1} after only three DeltaEdit edits across the five edited sites. The full table is in Appendix E.

5.5 Which parts of MoKE matter?

Table 4 isolates the context keys. On GPT2-XL and CounterFact, one key per edit gives 58.27 generalization. Seven context keys give 71.28 with no measurable loss in specificity at the reported precision. The gain is 5.30 points on Llama3-8B and CounterFact. On ZsRE, the difference is only 0.18 points. Multiple context keys therefore help in the setting where paraphrases are not well covered by one prompt representation. They are not a universal improvement.

The optimized value itself is more important. On GPT2-XL and CounterFact, 20 optimization steps give 98.93 efficacy. Five steps give 41.57 and one step gives 0.13. We also test the layer used for retrieval. Generalization is 71.10 at layer 17, 70.28 at layer 15, and 68.68 at layer 13. Finally, mean-centering the keys does not improve the overall tradeoff. It raises specificity by 2.23 points but lowers generalization by 12.26 points. Table 5 reports these checks.

Figure 3 shows the effect of the retrieval threshold. At $\tau = 0.95$, MoKE is too selective and generalization is 82.76 because many paraphrases do not retrieve their edit. Generalization reaches 90.88 at $\tau = 0.80$. Lowering the threshold further to 0.70 reduces efficacy by 9.1 points and generalization to 86.13. The threshold therefore needs to be selected rather than treated as an unimportant implementation setting. Appendix B gives a held-out selection procedure.

5.6 The remaining specificity loss is not explained by more cross-edit matches

As external memory grows, one natural explanation is that paraphrases increasingly retrieve a key belonging to a different edit. We instrument the retrieval step to measure this directly. Table 6 reports the fraction of paraphrase queries assigned to a key from another edit. Across a 15-fold increase in memory size, this fraction changes only from 44.80 to 45.43 percent, while specificity falls from 97.50 to 93.29. The measured increase in cross-edit matches is therefore too small to

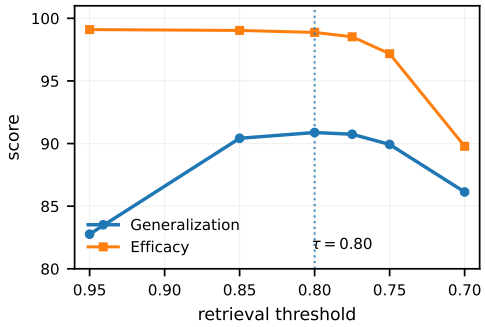

Figure 3: Threshold sweep on Llama3-8B and ZsRE after 3,000 edits. A high threshold causes paraphrases to miss their edit. Generalization is best at $\tau = 0.80$ and then falls when the threshold is lowered further.

Table 6: The specificity loss is not explained by more paraphrases being assigned to another edit. The measured rate is nearly flat across a 15x increase in memory size.

Measurement	Smaller memory	15x larger memory
Assigned to another edit	44.80%	45.43%
Specificity	97.50	93.29

Table 7: Variance and cost. Standard deviations use three MoKE seeds. Latency is greedy decoding, median of five 64-token runs.

Setting	σ Eff.	σ Gen.	σ Spe.
GPT2-XL / CF	.027	.980	.035
Llama3-8B / CF	.193	.598	.701
Llama3-8B / ZsRE	.091	.171	.000

Model	Keys	Latency	Memory
GPT2-XL	21k	+0.3%	269 MB
GPT2-XL	50k	+1.1%	640 MB
Llama3-8B	21k	+3.5%	688 MB
Llama3-8B	50k	+2.5%	1638 MB

Table 8: Numerical format on Llama3-8B and CounterFact after 3,000 edits. DeltaEdit changes substantially between bf16 and fp16, while MoKE changes little.

Method	Format	Eff.	Gen.	Spe.
DeltaEdit	bf16	49.37	29.85	21.80
DeltaEdit	fp16	93.57	63.32	34.21
MoKE	bf16	99.30	62.25	82.95
MoKE	fp16	99.03	63.05	83.98

explain the observed specificity loss. We also tested an attention-sink mask. It recovers part of the loss on GPT2-XL but has negligible effect on the two Llama3-8B settings. The cause of the remaining specificity decline is still unresolved.

5.7 Variance, Memory, and Inference Cost

We repeat the main MoKE settings with three seeds. The standard deviation of generalization is 0.980 on GPT2-XL and CounterFact, 0.598 on Llama3-8B and CounterFact, and 0.171 on Llama3-8B and ZsRE. These setting-specific values matter when interpreting small differences.

External memory also adds storage and retrieval cost. At the 3,000-edit operating point, the default CounterFact setup contains 21,000 keys and uses 269 MB for GPT2-XL and 688 MB for Llama3-8B. Measured decoding overhead is 0.3 and 3.5 percent respectively. At 50,000 keys, memory rises to 640 MB and 1.64 GB. Table 7 reports these measurements. Appendix F gives additional implementation details.

5.8 Repeated Weight Editing is Sensitive to Numerical Precision

Table 8 repeats Llama3-8B and CounterFact in bf16 and fp16 with the same code. DeltaEdit efficacy changes from 49.37 to 93.57, a difference of 44.20 points. MoKE changes from 99.30 to 99.03, a

difference of 0.27 points. Generalization and specificity show the same broad pattern. We do not claim that bf16 is generally unsuitable for weight editing. The result shows that numerical precision can dominate a sequential comparison when thousands of updates are accumulated in the same matrix.

Every controlled comparison in Table 2 therefore uses the same numerical format. We also withhold an AlphaEdit Llama3-8B and CounterFact fp16 run that did not match a published reference point. Validated AlphaEdit runs are in Appendix H.

5.9 What we mean by a capacity limit

The experiments do not imply one universal maximum number of edits. The limit depends on both the method and the edit distribution. CounterFact is the setting where moving the edit outside the weights helps most. ZsRE is the setting where the same change helps little or hurts, especially on Llama3-8B. This is why the paper studies capacity as an empirical scaling question rather than proposing a single edit-count bound.

The practical limits are also different. Repeated weight editing can fail sharply after a long stable region. MoKE keeps the base model fixed, but memory grows linearly and specificity gradually declines. Our measurements do not yet explain that decline.

6 Discussion

Why fixed representations help retrieval. If external memory were attached to a model that continued to receive weight edits, keys stored early and queries observed later would be produced by different versions of the network. Retrieval could then degrade because the representation itself had changed. MoKE avoids this source of drift by keeping the base model fixed.

What the 10,000-edit experiment does and does not show. The long-sequence result uses one backbone and one dataset. It does not establish a universal maximum number of edits. It shows that DeltaEdit can look stable at 2,500 to 3,000 edits and then collapse under the same sequential procedure, while MoKE behaves differently under that protocol. Longer edit sequences have been reported under other procedures (Yang et al., 2026; Gu et al., 2026; Dai et al., 2026). Direct ranking across those protocols would require rerunning them under a common setup.

When external memory is most useful. MoKE is not better in every setting. On Llama3-8B and ZsRE, DeltaEdit already preserves the edited facts well, and adding external memory reduces generalization. This suggests that external memory is most useful when repeated weight updates begin to interfere, rather than as a universal replacement for weight editing.

7 Conclusion

Sequential knowledge editing is usually treated as a problem of finding safer updates to model weights. Our experiments show that where the edit is stored also matters. Repeatedly changing the same parameters can work well for thousands of edits and then fail sharply as more updates accumulate. MoKE instead keeps the model frozen and stores the same optimized edit values in external memory. On CounterFact this change gives much better scaling and preserves general model behavior because the base weights never change. External memory is not a free solution. Its size grows linearly, the retrieval threshold must be calibrated, context keys help on CounterFact but not ZsRE, and specificity declines as more edits are stored. Llama3-8B on ZsRE is a clear case where DeltaEdit is already strong and MoKE makes generalization worse. We also tested whether the remaining specificity loss comes from paraphrases being increasingly matched to another edit. The measured rate is almost unchanged as memory grows, so that explanation is not supported. This also leaves an open question about what causes the gradual specificity loss in external memory as the edit sequence grows. The main conclusion is therefore conditional. Storing edits outside the weights can greatly extend sequential editing when repeated weight updates are the limiting factor, but it introduces a different set of costs that must also be measured.

ETHICS STATEMENT

This work studies methods for changing factual behavior in language models after training. The same mechanism can correct outdated information or install false information. External storage has one useful safety property in that individual edits are inspectable and removable without changing the base weights. It can also be distributed separately from the model, which creates its own misuse risk. We report specificity because it measures collateral changes to prompts that should remain unaffected. All experiments use public benchmarks and public model checkpoints. No human subjects or private data are used.

REPRODUCIBILITY STATEMENT

The method is specified by Equations 1 to 6 and the implementation details in Appendix A. We report model and dataset specific thresholds, numerical precision, seed variance, latency, memory, the held-out threshold selection procedure, and all main ablations. The abstract has link to the anonymized code repository for ease of reproducibility.

AI USE STATEMENT

Generative AI tools were used to provide feedback on experimental design, help interpret reported results, search and summarize related literature, and assist with drafting, editing, and LaTeX preparation. The experimental measurements reported in the paper were produced by the authors' code and checked by the authors. All AI-assisted text and organization were reviewed by the authors. The authors take responsibility for the final content.

REFERENCES

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference (TAC)*, 2009.

Yuchen Cai and Ding Cao. O-Edit: Orthogonal subspace editing for language model sequential editing. *arXiv preprint arXiv:2410.11469*, 2024.

Ding Cao, Yuchen Cai, Yuqing Huang, Xuesong He, Rongxi Guo, Guiquan Liu, and Guangzhong Sun. On the superimposed noise accumulation problem in sequential knowledge editing of large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. arXiv:2505.07899.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. Lifelong knowledge editing for LLMs with retrieval-augmented continuous prompt learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 13565–13580, 2024.

Yanbo Dai, Zhenlan Ji, Zongjie Li, and Shuai Wang. EAMET: Robust massive model editing via embedding alignment optimization. In *International Conference on Learning Representations (ICLR)*, 2026.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Xiaojie Gu, Ziying Huang, Jia-Chen Gu, and Kai Zhang. UltraEdit: Training-, subject-, and memory-free lifelong editing in language models. *Transactions on Machine Learning Research*, 2026. arXiv:2505.14679.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. Rebuilding ROME: Resolving model collapse during sequential model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 21738–21744, 2024a.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024b.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 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. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. WilKE: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *International Conference on Learning Representations (ICLR)*, 2023.

Houcheng Jiang, Junfeng Fang, Tianyu Zhang, Baolong Bi, An Zhang, Ruipeng Wang, Tao Liang, and Xiang Wang. Neuron-level sequential editing for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16678–16702, 2025.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

Qingyuan Liu, Jia-Chen Gu, Yunzhi Yao, Hong Wang, and Nanyun Peng. Energy-regularized sequential model editing on hyperspheres. In *International Conference on Learning Representations (ICLR)*, 2026.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022a.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022b.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. The butterfly effect of model editing: Few edits can trigger large language models collapse. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Wanli Yang, Rui Tang, Hongyu Zang, Du Su, Qi Cao, Jingang Wang, Huawei Shen, Xueqi Cheng, and Fei Sun. Fine-tuning done right in model editing. In *International Conference on Learning Representations (ICLR)*, 2026.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: Enhancing model editing with neuron-indexed dynamic LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 19449–19457, 2024.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Zhumin Chen, et al. Uncovering overfitting in large language model editing. In *International Conference on Learning Representations*, volume 2025, pp. 79550–79568, 2025.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

APPENDIX

A IMPLEMENTATION DETAILS

Models and data. We use GPT2-XL (Radford et al., 2019) and Llama3-8B (Grattafiori et al., 2024). The editing datasets are CounterFact (Meng et al., 2022) and ZsRE (Levy et al., 2017). The main protocol applies 3,000 edits one at a time. The long-sequence experiment uses GPT2-XL and CounterFact for 10,000 edits. DeltaEdit uses its released configuration and is not retuned specifically for the 10,000-edit experiment.

Edit layers and target values. We use the critical layers and target-value objective from the reference sequential editing configuration. GPT2-XL uses layers 13 through 17. Llama3-8B uses layers 4 through 8. MoKE retrieves at the last critical layer unless stated otherwise. The target value uses 20 optimization steps in the main configuration. DeltaEdit and MoKE use the same target-value objective.

Thresholds and keys. The main thresholds are 0.90 for GPT2-XL and CounterFact, 0.95 for GPT2-XL and ZsRE, 0.70 for Llama3-8B and CounterFact, and 0.80 for Llama3-8B and ZsRE. The default CounterFact memory uses seven context keys per edit. The GPT2-XL ZsRE configuration uses one key per edit. Similarities are computed between normalized hidden states.

Numerical precision. GPT2-XL experiments use fp32. Llama3-8B main runs use fp16. This choice matters for sequential weight editing. Section F shows a check where DeltaEdit changes by 44.2 efficacy points between bf16 and fp16 on Llama3-8B and CounterFact. MoKE changes by 0.27 points.

Metrics. Efficacy scores the edited prompt. Generalization scores paraphrases of the edit. Specificity scores unrelated neighborhood prompts and checks whether the original answer remains preferred. We use the top-token and larger-probability definitions from the adopted benchmark. The downstream GLUE test uses 200 examples per task and zero-shot accuracy. These accuracy values are not directly comparable with published F1 curves that use a different protocol.

B HELD-OUT THRESHOLD SELECTION

We test whether τ can be chosen without looking at evaluation edits. We split edits rather than prompts, so no paraphrase from a calibration edit appears in evaluation. Six hundred edits are used for calibration. For each candidate threshold, we compute the difference between the percentage of paraphrase queries that retrieve an edit and the percentage of neighborhood queries that retrieve an edit. We select the largest calibration difference and then keep the threshold fixed for evaluation.

The selected value is 0.875. Applying the same criterion directly to the evaluation edits also selects 0.875. The cost of choosing the threshold on held-out edits is therefore zero under this criterion. The headline GPT2-XL and CounterFact run uses 0.90, which was fixed before this held-out check. We report the held-out experiment as a selection procedure rather than changing the headline result after the fact.

Table 9: Held-out threshold selection on GPT2-XL and CounterFact. The threshold chosen on calibration is also the one selected on the evaluation split under the same criterion.

τ	Calibration difference	Evaluation difference
0.925	8.88	7.17
0.900	11.63	11.07
0.875	**12.33**	**11.75**
0.850	11.95	11.64
0.800	10.75	10.38

Table 10: Results at 10,000 edits on GPT2-XL and CounterFact. A second seed confirms the endpoint for both methods.

Method	Seed	Eff.	Gen.	Spe.
MoKE	0	97.81	69.81	82.45
MoKE	1	97.69	67.82	82.65
DeltaEdit	0	13.13	3.07	7.38
DeltaEdit	1	13.13	3.07	7.07

C Long-sequence replication and seed variance

The 10,000-edit endpoint was repeated for both DeltaEdit and MoKE. DeltaEdit reproduces 13.13 efficacy and 3.07 generalization to three decimals. MoKE remains above 97.6 efficacy on both runs. Specificity declines for both methods. The long-sequence experiment therefore supports gradual degradation for MoKE on this setting, not perfect preservation.

The Llama3-8B and ZsRE generalization loss in Table 1 is evaluated against its own standard deviation of 0.171. The 0.72-point gap to the best published value is about 4.2 standard deviations. Earlier drafts compared all cells with the GPT2-XL and CounterFact standard deviation of 0.980. We do not do that here.

D Component ablations

The target-value ablation gives the clearest result. With one optimization step, MoKE mostly reproduces the unedited model. This shows that MoKE depends on the optimized hidden value from the locate-then-edit procedure. Moving storage outside the weights does not replace that optimization.

The layer sweep uses the same attention-sink setting at every layer. The final critical layer gives the best generalization and specificity among the three tested layers. Mean-centering the keys changes the tradeoff rather than improving it. Specificity increases by 2.23 points while generalization falls by 12.26 points.

E Checking that the model weights remain unchanged

We measure the maximum absolute change in the edited weight matrices rather than assuming that the retrieval implementation leaves them untouched. DeltaEdit is used as a positive control for the same test.

All five MoKE matrices are bit-identical to the original checkpoint. The DeltaEdit control confirms that the same test detects weight changes when they occur.

Table 11: Maximum absolute change in the edited weights. MoKE is measured after 20 sequential edits. DeltaEdit is measured after three edits as a positive control.

Edited site	MoKE	DeltaEdit
h.13.mlp.c_proj	0.0	2.26×10^{-2}
h.14.mlp.c_proj	0.0	4.43×10^{-2}
h.15.mlp.c_proj	0.0	6.25×10^{-2}
h.16.mlp.c_proj	0.0	7.62×10^{-2}
h.17.mlp.c_proj	0.0	1.08×10^{-1}

Table 12: Numerical precision on Llama3-8B and CounterFact after 3,000 edits.

Method	Format	Eff.	Gen.	Spe.
DeltaEdit	bf16	49.37	29.85	21.80
DeltaEdit	fp16	93.57	63.32	34.21
MoKE	bf16	99.30	62.25	82.95
MoKE	fp16	99.03	63.05	83.98

F Numerical precision and cost

DeltaEdit is highly sensitive to numerical format in this setting. Thousands of updates are accumulated in the same matrices, so numerical precision is part of the experimental configuration. We therefore match precision in every controlled comparison.

The cost measurements in Table 12 use greedy generation of 64 tokens and report the median of five runs. The normalized external memory remains on the same device as the model. Moving and renormalizing the full memory on every forward pass is much slower and is not the implementation used for the reported latency. Memory use grows linearly with the number of stored keys.

G Comparison with earlier external-memory editors

GRACE (Hartvigsen et al., 2023) is the closest earlier method because it also stores hidden keys and values. Its published experiments do not include a model, dataset, and edit-count setting that overlaps our adopted 3,000-edit protocol. We therefore do not report an official GRACE rerun whose implementation cannot be checked against a published result. The single-key result in Table 4 is a controlled test of the shared key-value idea inside our implementation. It is not presented as a reproduction of GRACE. GRACE also differs in how its stored values and per-entry retrieval radii are learned.

WISE (Wang et al., 2024), SERAC (Mitchell et al., 2022b), Transformer-Patcher (Huang et al., 2023), and MELO (Yu et al., 2024) also keep editing state outside the original model parameters. Their evaluation protocols and mechanisms differ enough that copying a published row would not isolate the storage question studied here. Our main controlled baseline is therefore DeltaEdit run in the same implementation as MoKE.

H AlphaEdit checks in our implementation

We also run AlphaEdit (Fang et al., 2025) in our implementation when a published result is available for comparison. These are our runs rather than reproductions of the published table. Three settings score at or above the published AlphaEdit values, which provides a useful check that the implementation is not systematically weak.

The Llama3-8B and CounterFact fp16 adaptation did not match the published reference point and is therefore withheld. We do not use that failed setting in any comparison. On Llama3-8B and ZsRE, our AlphaEdit run reaches 95.27 generalization while MoKE reaches 90.80. This supports the main

Table 13: AlphaEdit in our implementation. Only settings that can be checked against a published result are reported.

	Our AlphaEdit			Published AlphaEdit		
Setting	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.
GPT2-XL / CF	92.40	56.10	53.19	85.93	46.23	53.18
GPT2-XL / ZsRE	93.77	85.77	25.08	93.19	84.97	23.55
Llama3-8B / ZsRE	98.23	95.27	29.88	94.49	91.52	29.77

negative result. External memory is not preferable when sequential weight editing is already highly stable.

I Negative results and unresolved behavior

Several ideas were tested and are not used as general claims.

More matches to another edit. The retrieval measurements do not support the simplest explanation for the loss in specificity. The fraction of paraphrase queries assigned to a key from another edit changes from 44.80 to 45.43 percent across a 15-fold increase in memory size, while specificity drops from 97.50 to 93.29. We therefore do not attribute the specificity decline to a growing rate of cross-edit matches.

Attention-sink masking. Masking the attention sink improves GPT2-XL and CounterFact specificity by 3.18 points, which explains part of the decline in that one setting. It is bit-identical on Llama3-8B and ZsRE and changes Llama3-8B and CounterFact specificity by only 0.36 points. It is not included as a general MoKE component.

Key geometry does not give a useful capacity formula. A packing calculation based on the effective key dimension and angular retrieval radius predicted a capacity of only about 14 edits in a setting that successfully stores 3,000 edits at 98.93 efficacy. The condition is too strong. We do not claim a geometric capacity theorem.

One threshold rule does not work across all settings. We tested a label-free rule based on the distribution of nearest competing-key similarities. The best threshold corresponds to very different quantiles across settings. A single quantile did not predict both. We therefore use the held-out calibration procedure in Appendix B.

J Extended related work

The view of transformer feed-forward layers as key-value memories helped motivate direct factual editing (Geva et al., 2021). Causal localization and the best place to intervene are not always the same (Hase et al., 2023). This distinction matters here because MoKE does not claim that the chosen hidden layer is the unique location where a fact is stored. It uses that layer because it is the same intervention point used by the reference editor.

Model editing evaluation has also expanded beyond success on the requested fact. Work on general capability loss, overfitting, and large-scale forgetting shows why unrelated behavior and long edit sequences need to be measured explicitly (Gu et al., 2024; Li et al., 2024; Zhang et al., 2025; Gupta et al., 2024b). Broader surveys cover a wide range of editing targets, methods, and evaluation settings (Yao et al., 2023). Our experiments focus on sequential factual editing because it provides a controlled setting for comparing repeated changes to shared weights with storage in separate external memory.