HEBBIANEDIT: RETHINKING KNOWLEDGE EDITING AS EXTERNAL MEMORY INJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge editing aims to update specific factual knowledge in a deployed language model without retraining. Existing methods rewrite the model's internal weights, train an extra module for every edit, or keep new facts as text outside the model. However, all three overlook that the model's knowledge itself lives in associative memory, giving up at least one of precision, training-free update, or revocability. In this work, we rethink knowledge editing as external memory injection, and propose HebbianEdit, which constructs an external writable memory, FactMLP, with all weights obtained in closed form. Specifically, we write new facts as key–value pairs into this memory, the key identifying an edit and the value carrying the new knowledge. Then we build the map between keys and values, whose quadratic layer realizes a squared key similarity that suppresses cross-talk, implemented as a two-layer MLP coupled with an activation gate. Finally, the frozen model reads the memory through hidden-state perturbation at inference time. Maintaining the deployed model's knowledge is thus achieved by routine memory operations rather than surgery per fact. Comprehensive experiments show that HebbianEdit sustains ten thousand sequential edits with efficacy at 96.5% or above at every scale on two benchmarks and three backbones. On CounterFact at ten thousand edits, the strongest weight editor retains only 73.2–86.0%. In addition, the proposed memory is built in a single closed-form pass in 1.8–14.4 minutes at ten thousand edits, compared to 6.6–11.0 hours of per-edit training for the strongest weight-preserving editor.

1 INTRODUCTION

Large language models (LLMs) (Kimi Team et al., 2026; Cao et al., 2026; GLM-5-Team et al., 2026) have revolutionized natural language processing in recent years, achieving remarkable performance across diverse tasks (Chen et al., 2021; Brown et al., 2020; Wei et al., 2022). Much of this success rests on factual knowledge absorbed into the parameters during pretraining through next-token prediction (Meng et al., 2022). Such knowledge, however, is frozen the moment training completes, while the world keeps producing new facts every day. A deployed model therefore quickly drifts out of date, and continuously retraining it is prohibitively expensive. Knowledge editing (Zhao et al., 2026; Meng et al., 2022; 2023; Hartvigsen et al., 2023; Wang et al., 2024) addresses this problem by updating specific factual knowledge in a deployed model while preserving unrelated knowledge, thereby avoiding the cost of full retraining.

Existing knowledge editing methods fall into three families (Figure 1(a)–(c)). **Weight-modifying** methods rewrite the weights that carry factual knowledge. ROME (Meng et al., 2022) locates the knowledge in an MLP layer and rewrites it with a rank-one update, and MEMIT (Meng et al., 2023) spreads such updates across layers to edit thousands of facts at once. **Weight-preserving** methods keep the base weights frozen and hold edits in external modules trained on the new facts, such as adapters (Yu et al., 2024) or codebooks (Hartvigsen et al., 2023). **Retrieval-augmented** methods (Lewis et al., 2020; Borgeaud et al., 2022) leave the model untouched and fetch new facts as text into the context at inference time. The families differ in where the new knowledge goes, and each placement has its cost. Weight-modifying methods write into the same weights that store everything else the model knows, so edits accumulate into catastrophic forgetting over long sequences (Yang et al., 2024), and no single edit can be revoked without restoring the original weights. Weight-preserving methods must learn the extra modules through training, a cost paid once per edit or per

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.

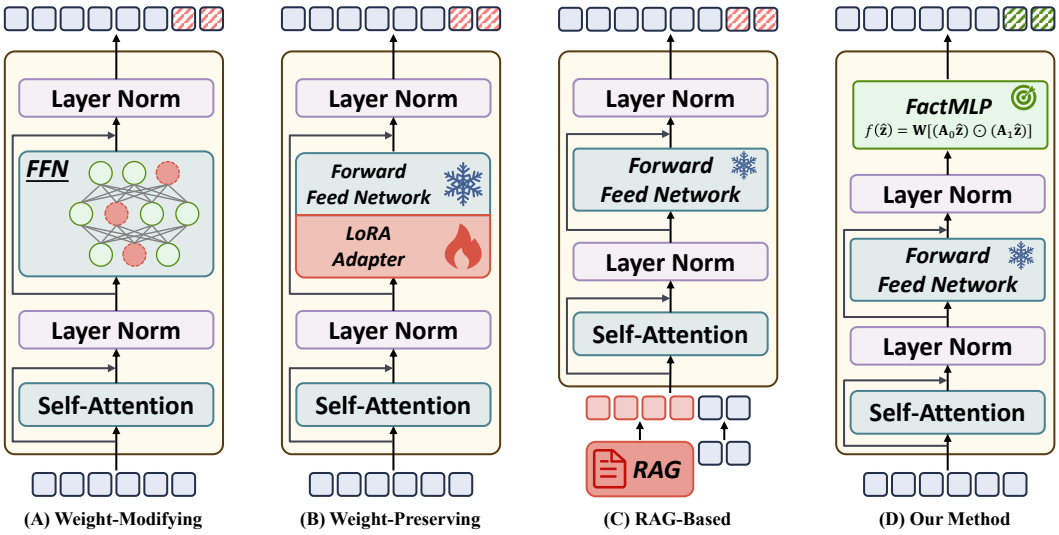


Figure 1: Knowledge editing methods differ in where the new knowledge goes. (a) Weight-modifying methods locate the neurons that carry factual knowledge and rewrite their weights. (b) Weight-preserving methods keep the weights frozen and train an extra module on the new facts. (c) Retrieval-augmented methods keep the new facts outside the model as text and retrieve them into the context at inference time. (d) HebbianEdit constructs an external memory, FactMLP, whose weights ($\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{W}$) are all obtained in closed form rather than trained, and plugs it into the frozen model, which reads it inside the forward pass.

batch of edits. Retrieval-augmented methods never make the new knowledge part of the model, i.e., the facts remain text in the context, and using them hinges on retrieval quality. None of the three families lets the model acquire new knowledge with precision, without training, and with revocability.

Neuroscience (Hebb, 1949; Marr, 1971) has long described knowledge as associative memory, or **Hebbian memory**, in which each entry pairs a cue with its content and returns the content when the cue is presented. Hopfield networks (Hopfield, 1982) gave the classical formalization of such memories, storing key–value pairs as a sum of outer products so that recall amounts to multiplying the memory matrix by a query. The principle recurs throughout sequence modeling, from fast-weight memories (Schmidhuber, 1992) and modern Hopfield attention (Ramsauer et al., 2021) to state space models that compress long contexts into a fixed-size memory (Gu & Dao, 2024). The same structure also sits inside pretrained Transformers, whose MLP layers, i.e., feed-forward networks (FFNs), serve as key–value associative memories that hold factual knowledge (Geva et al., 2021; Zhu et al., 2026) and were recently proven to be whitened Hebbian memories (Garcia et al., 2026). Weight editors have adopted the same view, writing new memories into these layers (Meng et al., 2022; 2023), and thus inherit the costs of a shared store, namely forgetting and irrevocability. Our insight is the complementary move. New knowledge should become the model's own new memory, written directly rather than learned, held in a separate store with the existing memory left fully intact.

We instantiate this insight in **HebbianEdit**, which constructs an external, writable memory, termed FactMLP, for new facts and plugs it into the frozen model. FactMLP couples an activation gate with a two-layer MLP, and all its weights are obtained in closed form in a single pass over the edits. Inside the memory, each fact is stored as a key–value pair. The key identifies the fact by its subject, and the value, built from the fact's tokens, carries the new knowledge. A squared key similarity suppresses cross-talk between the stored pairs. During inference, every token passes through FactMLP inside the forward pass, each presenting a key that queries the memory. When a key matches a stored edit, FactMLP perturbs the hidden state at the position that produces the next token, and the model recalls the new knowledge. When no key matches, the forward pass is left untouched. Existing knowledge thus remains intact, new facts are written rather than trained, memories from different domains are swappable, and every edit can be revoked precisely by deleting its entry.

Our contributions are threefold.

- We rethink knowledge editing as external memory injection. New knowledge should become the model's own new memory, written into a separate store outside the weights with the existing memory left intact, and read inside the forward pass. Under this view, the characteristic costs of existing methods trace to where the new knowledge is placed, and precision, training-free update, and revocability follow from the placement itself.
- We present HebbianEdit, which realizes this placement as FactMLP, an external writable memory coupling an activation gate with a two-layer MLP, whose closed-form weights give exact, per-entry insertion and deletion.
- We validate HebbianEdit at scale. It sustains ten thousand sequential edits with efficacy at 96.5% or above and the best generalization at every scale on two benchmarks and three backbones (Table 1). On CounterFact at ten thousand edits, its efficacy stays at 98.2% or above while the strongest weight editor retains only 73.2–86.0%, and the memory is built in a single closed-form pass over the edit set, in 1.8–14.4 minutes instead of 6.6–11.0 hours of per-edit training for the strongest weight-preserving editor (Table 6).

2 Preliminaries: Hebbian Memories

As introduced in Section 1, knowledge takes the form of associative memory. A Hebbian memory writes each key–value pair $(\boldsymbol{k}_i, \boldsymbol{v}_i)$ into a matrix as an outer product and stores the sum of all N pairs (Hopfield, 1982), so that recall with a query $\boldsymbol{q}$ is the sum of the stored values weighted by the query's inner product with each key,

$$\boldsymbol{M} \;=\; \sum_{i=1}^{N} \boldsymbol{v}_i\, \boldsymbol{k}_i^{\top}, \qquad \boldsymbol{M}\boldsymbol{q} \;=\; \sum_{i=1}^{N} (\boldsymbol{k}_i^{\top}\boldsymbol{q})\, \boldsymbol{v}_i. \tag{1}$$

The key–value structure gives editing a natural form. Every fact lives in its own entry addressed by its key, so adding, revising, or revoking a fact amounts to inserting, replacing, or deleting an entry, and since pairs are written independently, the whole memory is built in a single pass.

Stacking the keys and values as columns of two matrices $\boldsymbol{K}$ and $\boldsymbol{V}$ turns the recall of Eq. 1 into a two-layer MLP with identity activation, $\boldsymbol{M}\boldsymbol{q} = \boldsymbol{V}\boldsymbol{K}^{\top}\boldsymbol{q}$. Geva et al. (2021) find that the FFNs of pretrained Transformers are such memories with a nonlinear activation σ, i.e., $\mathrm{FFN}(\boldsymbol{x}) = \boldsymbol{V}\sigma(\boldsymbol{K}^{\top}\boldsymbol{x}) = \sum_i \sigma(\boldsymbol{k}_i^{\top}\boldsymbol{x})\, \boldsymbol{v}_i$. In this view, the first layer scores how strongly the input matches each key through a fixed similarity function and the second returns the activation-weighted sum of the values, while the keys and values themselves are learned from pretraining data. An editable memory thus exposes two degrees of freedom, the choice of the similarity function and the way the weights are obtained.

A central obstacle of Hebbian memories is the cross-talk of the additive readout. With normalized keys, recalling one fact by its key $\boldsymbol{k}_i$ returns its value plus a corruption from every other entry,

$$\boldsymbol{M}\boldsymbol{k}_i \;=\; \boldsymbol{v}_i \;+\; \sum_{j \neq i} (\boldsymbol{k}_j^{\top}\boldsymbol{k}_i)\, \boldsymbol{v}_j, \tag{2}$$

so other facts leak into the readout in proportion to their keys' similarity with $\boldsymbol{k}_i$, and the geometry of the key set, i.e., the pairwise similarities among all stored keys, determines how cleanly every fact can be recalled. We act on this corruption through both degrees of freedom, choosing a similarity that suppresses cross-talk in Section 3.3 and fitting the weights so that the memory returns each stored value at its own key in Section 3.4, while keeping the write closed form.

3 Method

We first formalize knowledge injection as hidden-state perturbation (Section 3.1), then construct the value and the key of an edit (Section 3.2), build FactMLP that gates and reads out the memory (Section 3.3), solve all its weights in closed form (Section 3.4), and describe how the built memory recalls a fact at inference (Section 3.5).

3.1 KNOWLEDGE INJECTION AS HIDDEN-STATE PERTURBATION

Let the base model be a frozen language model with hidden dimension d. An edit is a triple (S, R, F) of a subject, a relation, and a new fact, such as (France, president, Macron). After editing, the model answers F whenever S is asked about under the relation R, however the question is phrased.

To predict the next token, the model scores every token of the vocabulary against the hidden state after the final normalization layer, at the last position of the input sequence. Writing $\boldsymbol{h} \in \mathbb{R}^d$ for this hidden state and $\boldsymbol{v}_t \in \mathbb{R}^d$ for the unembedding vector of token t, the logit of t reads $\ell_t = \langle \boldsymbol{v}_t, \boldsymbol{h} \rangle$, and the highest-scoring token is produced.

We therefore formalize knowledge injection as hidden-state perturbation, i.e., adding a unit direction $\hat{\boldsymbol{y}}$ scaled by α to the hidden state so that every logit shifts by an exact amount,

$$\boldsymbol{h} \;\leftarrow\; \boldsymbol{h} + \alpha\, \hat{\boldsymbol{y}}, \qquad \ell'_t \;=\; \ell_t + \alpha\, \langle \boldsymbol{v}_t, \hat{\boldsymbol{y}} \rangle, \tag{3}$$

For an edit (S, R, F), the vector $\hat{\boldsymbol{y}}$ carries the new fact and must raise the fact's token above every competitor. We write the single-token case for simplicity. A fact may span several tokens, and its own perturbation then applies to each of them.

In practice, a deployed model is edited on a whole set $\{(S_i, R_i, F_i)\}_{i=1}^N$ of edits at once. The perturbation must therefore be selective, i.e., recognize which edit, if any, is being asked about and return its vector. This gives the memory a key–value form. We write the pair of edit i as $(\hat{\boldsymbol{z}}_i, \boldsymbol{y}_i)$. The key $\hat{\boldsymbol{z}}_i$ identifies edit i whenever its subject comes up, and the value $\boldsymbol{y}_i$ carries the new fact. Sections 3.2–3.4 construct the memory in closed form, and Section 3.5 describes how the built memory recalls a fact at inference.

3.2 CONSTRUCT THE VALUE AND KEY OF AN EDIT

Edit Values. By Eq. 3, the value of an edit must raise the logit of the fact's token above those of its competitors. Since a fact may span several tokens, we construct one value per fact position. For position j, write f_j for the fact's token at that position. Its competitors are the tokens that other facts place at position j, so we collect the unique tokens appearing at position j across all edits into the candidate set $\mathcal{V}_j$. The value is then the zero-centered unembedding of f_j,

$$\boldsymbol{y}_j \;=\; \boldsymbol{v}_{f_j} \;-\; \boldsymbol{\mu}_{\mathcal{V}_j}, \qquad \boldsymbol{\mu}_{\mathcal{V}_j} \;=\; \frac{1}{|\mathcal{V}_j|} \sum_{c \in \mathcal{V}_j} \boldsymbol{v}_c, \tag{4}$$

where $\boldsymbol{v}_c \in \mathbb{R}^d$ is the unembedding vector of candidate c. Zero-centering removes the component shared by all candidates, which would contribute equally to every candidate's logit. The memory stores $\boldsymbol{y}_j$, and injection uses its normalized version $\hat{\boldsymbol{y}}_j = \boldsymbol{y}_j / \|\boldsymbol{y}_j\|$, so that the strength is carried by α alone (Eq. 3). The logit gap between f_j and a competitor c then grows by $\alpha \langle \boldsymbol{y}_j - \boldsymbol{y}_c, \hat{\boldsymbol{y}}_j \rangle$ with $\boldsymbol{y}_c = \boldsymbol{v}_c - \boldsymbol{\mu}_{\mathcal{V}_j}$, which approaches $\alpha \|\boldsymbol{y}_j\|$ when the value is nearly orthogonal to the competitor vectors.

Edit Keys. The key must recognize its edit whenever the subject comes up, whatever the phrasing. At inference the model sees only the question, which excludes the fact, so to identify the edit we build the key from the subject and the relation only. Filling S into the template of R gives a statement such as *The president of France is*, and the key averages the normalized hidden states of the first l layers at the last token of the subject mention,

$$\hat{\boldsymbol{z}} \;=\; \mathrm{norm}\Big(\frac{1}{l} \sum_{\ell=0}^{l-1} \mathrm{norm}\big(\boldsymbol{h}^{(\ell)}\big) \Big), \qquad \mathrm{norm}(\boldsymbol{x}) = \boldsymbol{x} / \|\boldsymbol{x}\|, \tag{5}$$

with $\boldsymbol{h}^{(\ell)}$ denoting the hidden state of layer ℓ at that token, and $l = 4$ throughout the experiments (Appendix B). Rephrased queries about the same subject give nearby keys, since shallow layers encode entity identity (Appendix H).

3.3 FACTMLP: GATING AND READING OUT THE MEMORY

Recalling the fact of an edit takes two steps, gating, which decides whether a token activates a stored edit, and readout, which returns the value of that edit. The readout is a two-layer FFN, in which a

quadratic layer lifts the activated key into a bilinear feature and a readout layer maps the feature to the value direction. We introduce the activation gate and the two layers in turn (Figure 2).

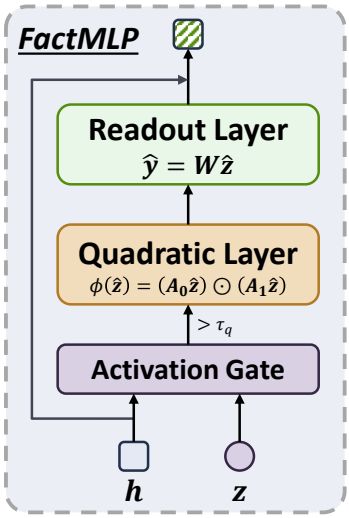


Figure 2: The FactMLP module (Section 3.3).

The Activation Gate. At inference, every token of the input produces a key $\hat{\boldsymbol{z}}_q$ by the construction of Eq. 5, with the hidden states taken at the token's own position. Stack the keys of all N edits as rows of a matrix $\boldsymbol{Z} \in \mathbb{R}^{N\times d}$. The gate scores each token's key against every stored key at once, $\boldsymbol{Z}\hat{\boldsymbol{z}}_q$, and lets the highest-scoring entry, with index i^*, through only if its score clears a single threshold τ_q (Appendix B). The winning entry's own stored key $\hat{\boldsymbol{z}}_{i^*}$, not the token's key, is then passed to the readout, so the readout always sees the inputs it was solved for. If no token's key clears τ_q, the gate stays closed, nothing is injected, and the model is left untouched.

The Quadratic Layer. The readout is a single map shared by all stored entries, so it must return the stored value at the target key and stay near zero at every other key. With the linear similarity $\langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}' \rangle$, a nearby non-target key would leak almost as much value as the target, while the unit keys give an exact match a score of one, which squaring leaves unchanged as it shrinks every other score to its square. Sharpening a similarity by a higher power is classical in associative memory (Krotov & Hopfield, 2016), and we use the square. The squared similarity admits an exact feature map with $\langle \phi(\hat{\boldsymbol{z}}), \phi(\hat{\boldsymbol{z}}') \rangle = \langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}' \rangle^2$, but its dimension $\frac{d(d+1)}{2}$ is prohibitive. The quadratic layer therefore approximates it with a compact bilinear feature,

$$\phi(\hat{\boldsymbol{z}}) \;=\; (\boldsymbol{A}_0\hat{\boldsymbol{z}}) \odot (\boldsymbol{A}_1\hat{\boldsymbol{z}}) \;\in\; \mathbb{R}^m, \tag{6}$$

with weights $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathbb{R}^{m\times d}$, which reduces the feature dimension to m. Appendix C proves that this approximation is accurate and gives an explicit error estimate. Writing $\rho = \langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}' \rangle$ for the correlation of two keys, the induced kernel $K(\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}') = \frac{1}{m}\langle \phi(\hat{\boldsymbol{z}}), \phi(\hat{\boldsymbol{z}}') \rangle$ concentrates on the squared correlation, $K(\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}') = \rho^2 + \eta$, where η is a zero-mean noise with variance at most $8/m$ at initialization, so the error vanishes as the width grows. We use $m = 2048$ throughout this paper, and Section 4.4 ablates the width.

The Readout Layer. Once the gate lets an entry through, the two layers map its stored key to the value direction,

$$\boldsymbol{y} \;=\; \boldsymbol{W}\left[(\boldsymbol{A}_0\hat{\boldsymbol{z}}_{i^*}) \odot (\boldsymbol{A}_1\hat{\boldsymbol{z}}_{i^*})\right], \tag{7}$$

where the readout layer $\boldsymbol{W} \in \mathbb{R}^{d\times m}$ reads out the stored values. It remains to determine the weights $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, and $\boldsymbol{W}$, which the next section solves in closed form. The edit stream is partitioned into consecutive shards of 5,000 edits. The gate scores each token's key against the stored keys of all shards, while the projections and readout of each shard are solved on its own keys (Appendix E).

3.4 Solving All Weights in Closed Form

The Quadratic Layer. The projections $(\boldsymbol{A}_0, \boldsymbol{A}_1)$ are neither random features (Rahimi & Recht, 2007; Garcia et al., 2026) nor learned by gradient descent. Starting from Gaussian draws, we refine each projection by a regularized least-squares objective that asks the memory's output at every stored key to align with its paired value. Appendix A gives the objective, the linear system it induces, and the derivation. The objective is a strictly convex quadratic in the projection, so its minimizer is unique and conjugate gradient returns it exactly, without forming the operator matrix. This is a solve of a determined system, not a training loop, since no data sampling, no learning rate, and no tuning beyond the regularizer are involved. We run a single round of alternation, first solving for $\boldsymbol{A}_1$ with $\boldsymbol{A}_0$ fixed and then for $\boldsymbol{A}_0$ with the refined $\boldsymbol{A}_1$ fixed, and Section 4.5 verifies that further rounds bring no gain.

The Readout Layer. With the quadratic layer fixed, the output weights $\boldsymbol{W}$ should reproduce the stored value at every stored key, so we fit them by regularized least squares against the stored values,

$$\boldsymbol{W} = \boldsymbol{Y}\boldsymbol{\Phi}^\top \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda_r \boldsymbol{I}\right)^{-1}, \tag{8}$$

Method		CounterFact									ZsRE								
		T = 300			T = 5000			T = 10000			T = 300			T = 5000			T = 10000		
		Eff.	**Gen.**	**Spe.**	**Eff.**	**Gen.**	**Spe.**	**Eff.**	**Gen.**	**Spe.**	**Eff.**	**Gen.**	**Spe.**	**Eff.**	**Gen.**	**Spe.**	**Eff.**	**Gen.**	**Spe.**
Pre-edited		0.30	0.20	15.0	0.30	0.50	13.9	0.30	0.40	14.0	31.6	32.3	34.8	32.3	32.6	35.1	32.1	32.2	34.8
FT	Qwen3-4B-Instruct	43.6	4.10	3.20	0.00	0.00	0.00	0.00	0.00	0.00	38.5	7.20	11.3	0.00	0.00	0.00	0.00	0.00	0.00
RLEdit	Qwen3-4B-Instruct	76.5	34.4	7.00	0.00	0.00	0.00	0.00	0.00	0.00	78.8	55.7	24.3	0.00	0.00	0.00	0.00	0.00	0.00
RECT	Qwen3-4B-Instruct	80.4	53.8	8.70	0.00	0.00	0.00	0.00	0.00	0.00	83.4	70.6	28.7	0.00	0.00	0.00	0.00	0.00	0.00
PRUNE	Qwen3-4B-Instruct	85.0	55.5	8.30	0.00	0.00	0.00	0.00	0.00	0.00	89.3	76.0	32.0	0.00	0.00	0.00	0.00	0.00	0.00
AdaEdit	Qwen3-4B-Instruct	88.4	59.5	8.10	0.00	0.00	0.00	0.00	0.00	0.00	85.0	73.0	28.4	0.00	0.00	0.00	0.00	0.00	0.00
SimIE	Qwen3-4B-Instruct	95.0	71.3	11.8	0.00	0.00	0.00	0.00	0.00	0.00	95.2	90.6	34.3	0.00	0.00	0.00	0.00	0.00	0.00
EMMET	Qwen3-4B-Instruct	98.5	74.0	11.7	0.00	0.00	0.00	0.00	0.00	0.00	98.5	93.2	36.8	0.00	0.00	0.00	0.00	0.00	0.00
PMET	Qwen3-4B-Instruct	98.8	72.4	12.6	82.0	55.2	7.20	0.00	0.00	0.00	98.6	94.0	37.3	88.2	80.3	35.5	0.00	0.00	0.00
MEMIT	Qwen3-4B-Instruct	97.0	70.8	13.2	85.3	58.3	8.10	0.00	0.00	0.00	96.5	93.0	37.1	95.5	89.9	38.9	90.0	81.2	36.3
AlphaEdit	Qwen3-4B-Instruct	98.5	74.1	12.8	0.00	0.00	0.00	0.00	0.00	0.00	97.9	93.3	37.0	89.1	84.0	37.4	0.00	0.00	0.00
BetaEdit†	Qwen3-4B-Instruct	98.0	73.5	13.0	90.3	60.1	8.00	78.5	54.8	3.80	97.5	93.0	36.7	96.7	90.9	38.1	92.1	84.0	37.0
HebbianEdit	Qwen3-4B-Instruct	**99.7**	**98.0**	**13.2**	**98.8**	**67.3**	**11.5**	**98.7**	**58.6**	**10.5**	**99.8**	**99.1**	32.7	**97.8**	**97.0**	35.9	**97.4**	**96.7**	**35.4**
Pre-edited		0.50	0.50	15.2	0.40	0.50	14.3	0.40	0.50	14.4	26.5	26.7	28.0	27.0	26.2	27.0	27.8	26.5	26.0
FT	GPT-J-6B	45.0	5.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	40.0	8.00	12.0	0.00	0.00	0.00	0.00	0.00	0.00
RLEdit	GPT-J-6B	80.0	36.0	5.50	0.00	0.00	0.00	0.00	0.00	0.00	81.0	58.0	19.0	0.00	0.00	0.00	0.00	0.00	0.00
RECT	GPT-J-6B	82.0	52.0	9.50	0.00	0.00	0.00	0.00	0.00	0.00	84.0	68.0	24.7	0.00	0.00	0.00	0.00	0.00	0.00
PRUNE	GPT-J-6B	87.0	54.0	10.6	0.00	0.00	0.00	0.00	0.00	0.00	88.0	74.0	25.6	0.00	0.00	0.00	0.00	0.00	0.00
AdaEdit	GPT-J-6B	89.0	58.0	10.0	0.00	0.00	0.00	0.00	0.00	0.00	86.0	71.0	24.8	0.00	0.00	0.00	0.00	0.00	0.00
SimIE	GPT-J-6B	96.4	72.1	12.8	0.00	0.00	0.00	0.00	0.00	0.00	94.5	88.0	27.5	0.00	0.00	0.00	0.00	0.00	0.00
EMMET	GPT-J-6B	98.8	73.0	13.5	0.00	0.00	0.00	0.00	0.00	0.00	98.0	92.0	28.8	94.3	87.5	26.7	87.5	82.6	22.8
PMET	GPT-J-6B	98.5	72.5	13.5	85.0	50.0	7.80	30.0	18.0	2.50	98.0	93.0	29.4	89.0	81.0	28.0	0.00	0.00	0.00
MEMIT	GPT-J-6B	97.8	71.0	13.8	74.5	40.4	7.60	27.2	15.5	2.30	96.5	92.0	30.1	95.0	89.0	24.0	0.00	0.00	0.00
AlphaEdit	GPT-J-6B	98.9	73.5	13.7	95.9	50.9	6.90	82.6	40.3	3.90	98.0	94.0	29.5	96.5	92.0	26.0	88.3	83.1	22.8
BetaEdit†	GPT-J-6B	98.8	74.0	13.9	96.6	52.8	6.80	86.0	41.8	4.10	97.5	93.0	29.6	98.9	95.1	26.7	92.5	86.5	24.7
HebbianEdit	GPT-J-6B	**99.7**	**97.3**	13.3	**99.0**	**82.4**	**10.3**	**98.8**	**45.6**	**10.9**	**99.5**	**98.7**	25.9	**99.0**	**98.2**	26.2	**98.9**	**98.0**	**25.8**
Pre-edited		0.40	0.20	22.3	0.30	0.30	21.6	0.30	0.40	21.8	36.5	36.9	37.5	37.2	36.6	38.5	37.8	37.4	37.2
FT	LLaMA3-8B-Instruct	48.0	5.90	8.70	0.00	0.00	0.00	0.00	0.00	0.00	42.0	8.00	12.0	0.00	0.00	0.00	0.00	0.00	0.00
RLEdit	LLaMA3-8B-Instruct	80.0	38.0	14.7	0.00	0.00	0.00	0.00	0.00	0.00	82.0	60.0	28.0	0.00	0.00	0.00	0.00	0.00	0.00
RECT	LLaMA3-8B-Instruct	84.0	58.0	17.0	0.00	0.00	0.00	0.00	0.00	0.00	86.0	74.0	32.0	0.00	0.00	0.00	0.00	0.00	0.00
PRUNE	LLaMA3-8B-Instruct	88.0	60.0	17.5	0.00	0.00	0.00	0.00	0.00	0.00	90.0	80.0	36.0	0.00	0.00	0.00	0.00	0.00	0.00
AdaEdit	LLaMA3-8B-Instruct	90.0	63.0	18.5	0.00	0.00	0.00	0.00	0.00	0.00	88.0	76.0	32.0	0.00	0.00	0.00	0.00	0.00	0.00
SimIE	LLaMA3-8B-Instruct	96.0	68.0	20.7	0.00	0.00	0.00	0.00	0.00	0.00	96.5	92.0	36.0	0.00	0.00	0.00	0.00	0.00	0.00
EMMET	LLaMA3-8B-Instruct	98.5	69.4	20.2	0.00	0.00	0.00	0.00	0.00	0.00	98.5	94.0	40.5	0.00	0.00	0.00	0.00	0.00	0.00
PMET	LLaMA3-8B-Instruct	98.8	63.0	21.6	82.0	60.3	12.8	0.00	0.00	0.00	98.8	94.5	39.4	91.0	87.0	41.0	80.1	74.3	37.5
MEMIT	LLaMA3-8B-Instruct	99.3	61.0	21.5	87.1	65.4	14.4	0.00	0.00	0.00	98.5	94.0	39.5	97.0	93.1	43.5	93.2	89.3	44.2
AlphaEdit	LLaMA3-8B-Instruct	97.3	71.5	19.4	0.00	0.00	0.00	0.00	0.00	0.00	86.0	81.0	40.1	84.2	79.0	39.3	0.00	0.00	0.00
BetaEdit†	LLaMA3-8B-Instruct	96.8	70.4	19.0	89.0	62.4	15.1	73.2	57.4	10.2	98.0	94.5	40.3	97.4	93.2	44.2	96.6	92.4	43.9
HebbianEdit	LLaMA3-8B-Instruct	**99.7**	**100**	20.5	**98.7**	**92.0**	**19.3**	**98.2**	**80.1**	**18.5**	**99.7**	**99.0**	**38.0**	97.0	**96.4**	**38.3**	96.5	**96.0**	38.0

Table 1: Editing performance across 300, 5,000, and 10,000 sequential edits on COUNTERFACT and ZSRE. Higher values are better for all metrics (efficacy, generality and specificity). Baseline rows are reproduced verbatim from Liu et al. (2026) († marks the namesake editor). The HebbianEdit rows are ours, run in the same harness with the same splits and scoring on the frozen backbone. Bold marks the best row of each column. For specificity, best means the smallest deviation from the block's pre-edited row.

where $\boldsymbol{Y}$ stacks the stored values and $\boldsymbol{\Phi}$ the features of the stored keys as columns, and λ_r is the regularizer. Like the quadratic layer, this is a single algebraic step with no gradient descent and no tuning beyond λ_r. Facts that span several tokens share the refined projections, and one readout layer is solved per fact position, with $\boldsymbol{Y}$ stacking the values at that position across all edits.

The normal equations behind this construction share their least-squares skeleton with batch weight editors such as MEMIT (Meng et al., 2023), but what is solved and where it lives differ. They fit a weight update fused into the model's MLP, while we fit the projections and readout of a module that stays outside the weights. All three weights of FactMLP are now determined, and each shard is built in a single closed-form pass over its edits. Each edit occupies one entry of the memory, so deleting an entry revokes its edit, detaching the memory restores the frozen model exactly, and memories of different domains are swappable.

3.5 RECALLING A FACT AT INFERENCE

At inference the memory rides inside the forward pass of the frozen model, and generation repeats the same pass for every new token. As soon as the key of some token clears the activation gate

(Section 3.3), the memory injects at the position where the model produces the next token, the last non-pad position of the input. The stored key of the selected entry passes through the quadratic and readout layers of its shard, and the output is normalized and added to the hidden state after the final normalization layer with strength α, so every logit shifts by exactly the amount of Eq. 3. We use $\alpha = 256$ throughout the experiments (Appendix B), and Appendix H sweeps the strength together with the injection site. The strength does not decay with the match score, since the gate has already voted, and discounting the injection would penalize an unseen phrasing twice.

Recall is transient. Once the edited model speaks the fact, the subject no longer appears in the freshly generated text, so the gate closes and the following tokens are produced exactly as in the frozen model. When no token clears the gate, nothing is injected, and the forward pass is bit-identical to the frozen model's.

4 Experiments

4.1 Setup

We follow the BetaEdit evaluation protocol (Liu et al., 2026) and its harness, with three metrics. Efficacy measures whether the model gives the edited fact after each edit, generality whether it gives the fact under paraphrased queries, and specificity whether unrelated facts remain unchanged. Evaluation runs on CounterFact (Meng et al., 2022) and zsRE (Levy et al., 2017) with three instruction-tuned backbones, Qwen3-4B-Instruct-2507 (Yang et al., 2025), GPT-J-6B (Wang & Komatsuzaki, 2021), and LLaMA3-8B-Instruct (Grattafiori et al., 2024). The mechanism analyses (Sections 4.3–4.5) use the production keys of Qwen3-4B-Instruct-2507 on 10,000 CounterFact edits (Section 4.5 uses one 5,000-key shard of this pool), stored raw as constructed, and the capacity measurements average over 24 random seeds. The ablations run on 5,000 edits. This is one production shard: construction shares no parameters across shards and component-level conclusions transfer to any stream length. The recall gate is the only cross-shard element and Appendix E shows it is scale-flat. One hyperparameter set is frozen across all experiments (Appendix B), and decoding is a single greedy pass at temperature zero.

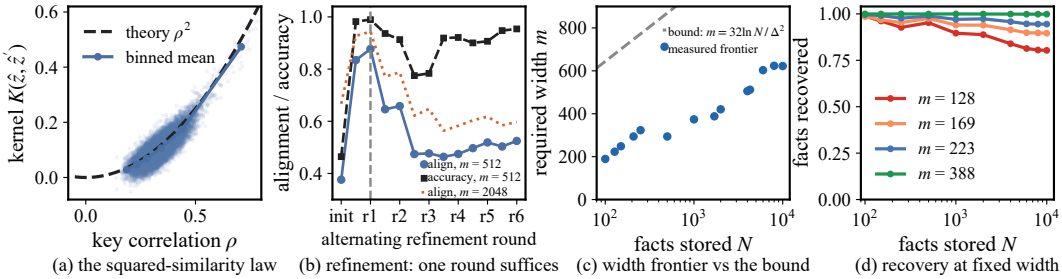
4.2 Sequential editing at scale

Long sequences of successive edits are where writing into shared weights accumulates interference with everything already stored. Table 1 reports editing performance under 300, 5,000, and 10,000 sequential edits on three backbones, against the editors shipped with the BetaEdit harness (Meng et al., 2022; 2023; Gupta et al., 2024b; Fang et al., 2025; Liu et al., 2026). On CounterFact, HebbianEdit keeps efficacy at 98.2 or above at every scale on every backbone, while the strongest weight editor retains only 78.5, 86.0, and 73.2 at ten thousand edits. Generalization at ten thousand edits reads 58.6, 45.6, and 80.1 on the three backbones, all above the strongest editor's 54.8, 41.8, and 57.4, and specificity there reads 10.5, 10.9, and 18.5 against pre-edited 14.0, 14.4, and 21.8 and against the strongest editor's 3.8, 4.1, and 10.2. On zsRE, efficacy stays at 96.5 or above and generalization at 96.0 or above at every scale, and specificity tracks the pre-edited model within 2.1 points everywhere, exceeding it by 0.8 and 0.6 points on Qwen3-4B-Instruct-2507 at 5,000 and 10,000 edits. Long sequences are exactly where a shared store must fail and an external memory has nothing to interfere with, so this gap is structural. Construction is a single closed-form pass over the edit set.

4.3 The squared-similarity law

The squared-similarity law $K(\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}') = \rho^2 + \eta$ with $\operatorname{Var}\eta \le 8/m$ (Appendix C) contains one informal step in its derivation, the small-deviation approximation on the noise η, so we measure the kernel directly. The pool consists of the production keys of Section 4.1. Their pairwise background correlation concentrates in $\rho \in [0.25, 0.4]$, a heavy background inherited from the anisotropy of the hidden-state space. We draw about 50,000 ordered pairs from the pool, compute the bilinear features of Eq. 6 from Gaussian projections at the operating width $m = 2048$, take the measured kernel of a pair as the scaled feature inner product $\frac{1}{m}\langle\phi(\hat{\boldsymbol{z}}), \phi(\hat{\boldsymbol{z}}')\rangle$, and bin the pairs by their key correlation.

Figure 3(a) plots the measured kernel against the key correlation. The binned means track the ρ^2 curve across the full range and sit slightly below it. At $\rho = 0.278$ the measured kernel reads 0.068



(a) the squared-similarity law (b) refinement: one round suffices (c) width frontier vs the bound (d) recovery at fixed width

Figure 3: Mechanism and capacity of the bilinear feature. (a) The measured kernel tracks the squared-similarity law. (b) One round of alternating refinement suffices. (c) The measured capacity frontier is linear in $\ln N$ and stays inside the bound. (d) Recovery at fixed width degrades gradually past the frontier.

against a theoretical 0.077, at 0.344 it reads 0.106 against 0.118, at 0.439 0.175 against 0.193, and at 0.708 0.474 against 0.502. The suppression matters where the mass is. Background pairs at $\rho \in [0.25, 0.4]$ drop to kernels between 0.06 and 0.16, so cross-talk between stored facts is second-order small. The bilinear feature therefore realizes the squared similarity with negligible error at the operating width.

4.4 Width scales logarithmically with fact count

The kernel concentration turns into a capacity bound through a union bound over the N^2 ordered key pairs, giving $m \geq 32 \ln N / \Delta^2$ (Appendix C). Union bounds can be loose, so we measure the actual frontier on the production key pool. The grid covers fact counts N up to 10,000, sixteen widths $m \in [128, 8192]$, and 24 seeds per cell. Each pool determines its own margin $\Delta = 1 - \rho^{*2}$ from the 99.9th percentile of its pairwise correlations, which stays between 0.42 and 0.49 with representative value 0.478. An entry fails when its kernel noise crosses the margin of its pool, and the frontier is the crossing where the expected failure count over the 24 seeds reaches one. Sweeping N at fixed width we interpolate linearly in $\ln N$, and sweeping width at fixed N we interpolate linearly, the two directions together giving fourteen frontier points.

Figure 3(c) plots the fourteen measured frontier points against the bound at the representative margin $\Delta = 0.478$. The frontier is linear in $\ln N$ with $R^2 = 0.936$, so the required width indeed grows logarithmically with the fact count. The measured slope of 87.6 per unit $\ln N$ sits below the bound's slope of 139.9 because the bound replaces the whole key population by its worst pair, and every point lies inside the bound.

Figure 3(d) shows the complementary view at fixed width. Recovery degrades gradually past the frontier rather than collapsing: at $N = 10{,}000$ the width $m = 128$ still recovers about 80% of the facts and $m = 388$ recovers over 99%. These curves describe a single memory. The production system shards the edit stream at 5,000 edits (Appendix E) and every solve operates deep inside the reliable region at the operating width $m = 2048$.

4.5 Alternating refinement converges in one round

Section 3.4 solves each projection with a single round of alternation rather than iterating to convergence, keeping the construction a single closed-form pass. This shortcut is sound only if one round recovers essentially all achievable readout quality, and we verify this on one production shard of 5,000 keys from the Qwen3-4B-Instruct-2507 CounterFact pool. Starting from Gaussian projections, we alternate closed-form solves of Eq. 10, one projection at a time with the other fixed, and after every half-step we fit a fresh ridge readout of Eq. 8 and record the mean cosine alignment between the readout at each stored key and its stored value, together with the argmax accuracy over the stored values.

Figure 3(b) shows the trajectory at the production shard size. At the operating width $m = 2048$ the random projections reach 0.667 alignment and 0.948 accuracy. The first $\boldsymbol{A}_1$ solve lifts them to 0.927 and 0.994 and the following $\boldsymbol{A}_0$ solve reaches 0.944 and 0.995, the production stop marked by the dashed line. Further rounds decay to 0.60 alignment by round six. The scarcer width $m = 512$ starts

at 0.376 and 0.464 and peaks at 0.877 and 0.989 after the same single round before decaying. One round therefore recovers all achievable alignment at both widths, and the construction stops after the first $\boldsymbol{A}_0$ solve.

5 Related Work

Under the memory view, placement decides cost. We re-examine the three families of Section 1 along this axis and close with HoReN and Larimar, the two closest works to our own.

Weight-modifying methods write the new knowledge into the weights of the backbone directly, from simple finetuning on the new facts to the locate-then-edit updates of Section 1 (Meng et al., 2022; 2023; Gupta et al., 2024b). Other lines predict the update with a hypernetwork (De Cao et al., 2021; Mitchell et al., 2022a; Tan et al., 2024) or constrain where it may go, into localized neurons (Pan et al., 2025) or the null space of preserved knowledge (Fang et al., 2025). Under the memory view, this family writes into the one store that also holds every pretrained fact, and the measured failures follow the placement. A few writes can collapse the model (Yang et al., 2024), later writes erode earlier ones as the sequence grows (Gupta et al., 2024a; Li et al., 2024), and undoing a write means rolling the weights back. HebbianEdit instead keeps the existing memory intact and stores new knowledge outside the weights.

Weight-preserving methods leave the backbone's weights untouched and store the edits in auxiliary modules, whether calibration slots or codebooks (Dong et al., 2022; Hartvigsen et al., 2023), LoRA blocks trained per batch of edits (Yu et al., 2024), or side modules selected by a learned router (Wang et al., 2024; Mitchell et al., 2022b). These modules are forms of memory, but they are learned rather than written. Every entry comes from a round of gradient training, the routers are trained up front, and revising a stored entry requires another round. HebbianEdit instead constructs the memory in closed form, so each entry is computed directly and can be replaced individually.

Retrieval-augmented methods keep the new facts outside the model and bring them in as text, either by reading retrieved passages into the context (Lewis et al., 2020; Borgeaud et al., 2022; Shi et al., 2024; Khandelwal et al., 2020; Asai et al., 2024) or by editing through in-context demonstrations and feedback (Zheng et al., 2023; Madaan et al., 2022). Such methods can inject knowledge effectively (Ovadia et al., 2024), but the facts stay text outside the parameters, and each answer is bounded by what the retriever returns and how the context is used (Chen et al., 2024; Liu et al., 2024). HebbianEdit instead writes the knowledge into FactMLP, and the model reads it during the forward pass.

In parallel to our work, HoReN (Fang et al., 2026) also brings an associative memory to knowledge editing by wrapping one MLP layer, but its values are a learned codebook, and an edit is retrieved by angular similarity. HebbianEdit differs on both axes. The memory is built directly from the given facts in closed form, not learned as a codebook. Its readout is a continuous map applied through hidden-state perturbation, so the returned value is computed rather than fetched. Larimar (Das et al., 2024) likewise endows a model with an episodic memory, but its reading and writing machinery is trained into the architecture, while HebbianEdit attaches to a frozen deployed model.

6 Conclusion

This paper rethinks knowledge editing as external memory injection. A model's knowledge is organized as associative memory, so HebbianEdit leaves the pretrained memory intact and writes new facts as key–value pairs into FactMLP, whose weights are all solved in closed form. It sustains ten thousand sequential edits with efficacy at 96.5% or above at every scale, and detaching the memory restores the frozen model exactly. The frozen memory from pretraining and the writable external memory after deployment then play complementary roles, and maintaining a deployed model's knowledge becomes a routine memory operation rather than surgery per fact.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Confer-*

ence on Learning Representations, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, Zeyao Ma, Kashun Shum, Xuwu Wang, Jinxi Wei, Jiaxi Yang, Jiajun Zhang, Lei Zhang, Zongmeng Zhang, Wenting Zhao, and Fan Zhou. Qwen3-coder-next technical report, 2026. URL https://arxiv.org/abs/2603.00729.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29728. URL https://doi.org/10.1609/aaai.v38i16.29728.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Payel Das, Subhajit Chaudhury, Elliot C. Nelson, Igor V. Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie C. Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiří Navrátil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control. In *Proceedings of ICML*, 2024.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.522. URL https://aclanthology.org/2021.emnlp-main.522/.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. Calibrating factual knowledge in pretrained language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang

(eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 5937–5947, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.438. URL https://aclanthology.org/2022.findings-emnlp.438/.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HvSytvg3Jh.

Yuan Fang, Yi Xie, and Xuming Ran. Horen: Normalized hopfield retrieval for large-scale sequential model editing, 2026. URL https://arxiv.org/abs/2605.08143.

Roberto Garcia, Jerry Weihong Liu, Ronald Guenther Junkins, Owen M Dugan, Dylan Zinsley, Sabri Eyuboglu, Atri Rudra, and Christopher Re. MLPs are hebbians: Constructing efficient fact-storing MLPs for transformers. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=fZd6X8gTRw.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary,

Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa,

Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15202–15232, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.902. URL https://aclanthology.org/2024.findings-acl.902/.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. A unified framework for model editing. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15403–15418, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.903. URL https://aclanthology.org/2024.findings-emnlp.903/.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: lifelong model editing with discrete key-value adaptors. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Donald O. Hebb. *The Organization of Behavior*. Wiley, 1949.

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554. URL https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng

Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Dmitry Krotov and John J. Hopfield. Dense associative memory for pattern recognition. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 1180–1188, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia (eds.), *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL https://aclanthology.org/K17-1034/.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

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 *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Bingqing Liu, Wei Liu, and Yuhua Li. Betaedit: Null-space constrained sequential model editing. In Diego Calvanese (ed.), *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence, IJCAI-26*, pp. 4563–4571. International Joint Conferences on Artificial Intelligence Organization, 8 2026. doi: 10.24963/ijcai.2026/508. URL https://doi.org/10.24963/ijcai.2026/508. Main Track.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve GPT-3 after deployment. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2833–2861, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.183. URL https://aclanthology.org/2022.emnlp-main.183/.

David Marr. Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 262(841):23–81, 1971. doi: 10.1098/rstb.1971.0078.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of ICML*, 2022b.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Fine-tuning or retrieval? comparing knowledge injection in LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 237–250, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.15. URL https://aclanthology.org/2024.emnlp-main.15/.

Haowen Pan, Xiaozhi Wang, Yixin Cao, Zenglin Shi, Xun Yang, Juanzi Li, and Meng Wang. Precise localization of memories: A fine-grained neuron-level knowledge editing technique for LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5xP1HDvpXI.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'07, pp. 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tL89RnzIiCd.

Jürgen Schmidhuber. Learning to control fast-weight memories: an alternative to dynamic recurrent networks. *Neural Comput.*, 4(1):131–139, January 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.131. URL https://doi.org/10.1162/neco.1992.4.1.131.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=L6L1CJQ2PE.

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

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 *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5419–5437, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.322. URL https://aclanthology.org/2024.findings-acl.322/.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29916. URL https://doi.org/10.1609/aaai.v38i17.29916.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Yiwen Hu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *Frontiers of Computer Science*, 20(12), May 2026. ISSN 2095-2236. doi: 10.1007/s11704-026-60308-3. URL http://dx.doi.org/10.1007/s11704-026-60308-3.

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 Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4862–4876, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.296. URL https://aclanthology.org/2023.emnlp-main.296/.

Jinchang Zhu, Jindong Li, Cheng Zhang, Jiahong Liu, and Menglin Yang. HeLa-mem: Hebbian learning and associative memory for LLM agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13757–13769, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.625. URL https://aclanthology.org/2026.acl-long.625/.

A Derivation of the refinement system

Section 3.4 solves each projection, with the other one fixed, by regularized least squares. Writing $\boldsymbol{C} = \boldsymbol{A}_0\boldsymbol{Z}^\top$ for the fixed side with $\boldsymbol{Z}$ stacking the stored keys as rows, $\boldsymbol{P} = \boldsymbol{A}_1$ for the projection being solved (the roles swap for the $\boldsymbol{A}_0$ solve), and $\boldsymbol{W}^0$ for the readout of Eq. 8 fit at the current projections, the objective asks that the output at each stored key match its paired value and no other stored value,

$$\min_{\boldsymbol{P}} \ \big\|\boldsymbol{W}^0\big(\boldsymbol{C}\odot\boldsymbol{P}\boldsymbol{Z}^\top\big)\big\|_{\boldsymbol{G}}^2 - 2\big\langle\boldsymbol{W}^0\big(\boldsymbol{C}\odot\boldsymbol{P}\boldsymbol{Z}^\top\big),\boldsymbol{Y}\big\rangle_F + \lambda_r\|\boldsymbol{P}\|_F^2, \qquad \boldsymbol{G} = \boldsymbol{Y}\boldsymbol{Y}^\top, \tag{9}$$

where $\|\boldsymbol{X}\|_{\boldsymbol{G}}^2 = \mathrm{tr}(\boldsymbol{X}^\top\boldsymbol{G}\boldsymbol{X})$. This appendix derives its minimizer. Abbreviate

$$\boldsymbol{F}(\boldsymbol{P}) \ = \ \boldsymbol{C}\odot\boldsymbol{P}\boldsymbol{Z}^\top, \qquad \boldsymbol{X}(\boldsymbol{P}) \ = \ \boldsymbol{W}^0\boldsymbol{F}(\boldsymbol{P}),$$

so the objective reads

$$\min_{\boldsymbol{P}} \ \|\boldsymbol{X}(\boldsymbol{P})\|_{\boldsymbol{G}}^2 - 2\langle\boldsymbol{X}(\boldsymbol{P}),\boldsymbol{Y}\rangle_F + \lambda_r\|\boldsymbol{P}\|_F^2,$$

where the first term sums the squared alignments of the outputs with all stored values because $\|\boldsymbol{X}\|_{\boldsymbol{G}}^2 = \|\boldsymbol{Y}^\top\boldsymbol{X}\|_F^2$. The gradient of this objective with respect to $\boldsymbol{X}$ is $2\boldsymbol{G}\boldsymbol{X} - 2\boldsymbol{Y}$. Passing this derivative through $\boldsymbol{X} = \boldsymbol{W}^0\boldsymbol{F}$ multiplies it by $\boldsymbol{W}^{0\top}$, passing it through the elementwise product $\boldsymbol{F} = \boldsymbol{C}\odot\boldsymbol{P}\boldsymbol{Z}^\top$ multiplies it pointwise by $\boldsymbol{C}$, and passing it through the linear map $\boldsymbol{P}\mapsto\boldsymbol{P}\boldsymbol{Z}^\top$ right-multiplies by $\boldsymbol{Z}$. Setting the resulting gradient to zero and dividing by two gives the linear system of Eq. 10,

$$\big[\boldsymbol{C}\odot\big(\boldsymbol{W}^{0\top}\boldsymbol{G}\boldsymbol{W}^0(\boldsymbol{C}\odot\boldsymbol{P}\boldsymbol{Z}^\top)\big)\big]\boldsymbol{Z} + \lambda_r\boldsymbol{P} \ = \ \big(\boldsymbol{C}\odot\boldsymbol{W}^{0\top}\boldsymbol{Y}\big)\boldsymbol{Z}. \tag{10}$$

With $\lambda_r > 0$ this quadratic is strictly convex in $\boldsymbol{P}$, so the system is symmetric positive definite with a unique solution. Each application of the operator on the left requires only matrix products and elementwise multiplications, so we solve it with conjugate gradient, never assembling the operator.

	CounterFact			zsRE	
backbone	300	5,000	10,000	300	5,000/10,000
GPT-J-6B	0.83	0.92	0.965	0.92	0.94
LLaMA3-8B	0.75	0.95	0.975	0.90	0.90
Qwen3-4B	0.90	0.97	0.975	0.90	0.97

Table 2: The frozen gate threshold τ_q of each backbone–dataset–stream-length cell. zsRE keeps one threshold across the two larger scales on every backbone.

B HYPERPARAMETERS

One hyperparameter set is frozen across all datasets and all model sizes. The key averages the hidden states of layers 0–3 (Eq. 5). The readout and refinement regularizer is $\lambda_r = 10^{-6}$ (Eqs. 10 and 8). The injection strength is $\alpha = 256$ (Eq. 3). The feature width is $m = 2048$ throughout (Section 3.3), and the width scans of Section 4.4 cover $m \in [128, 8192]$. Each memory is built in consecutive shards of 5,000 edits and each shard is solved independently (Section 3.3). Construction runs in float64. The gate threshold τ_q is the only inference-time hyperparameter. Its value is fixed per backbone–dataset–stream-length cell (Table 2). All runs are single greedy passes at temperature zero.

C CAPACITY DERIVATION

This appendix derives the concentration behavior of the bilinear feature $\phi(\hat{\boldsymbol{z}}) = (\boldsymbol{A}_0\hat{\boldsymbol{z}}) \odot (\boldsymbol{A}_1\hat{\boldsymbol{z}})$ with unit-norm keys, where the rows of $\boldsymbol{A}_0, \boldsymbol{A}_1$ are drawn i.i.d. from $\mathcal{N}(0, \boldsymbol{I}_d)$, as at initialization before refinement, in the spirit of random-feature kernel sketches (Rahimi & Recht, 2007). The main result is the kernel concentration of Eq. 11.

Kernel concentration. Fix two keys $\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}'$ with $\rho = \langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}' \rangle$. For one component r, write $u_b = \boldsymbol{a}_{b,r}^\top \hat{\boldsymbol{z}}$ and $v_b = \boldsymbol{a}_{b,r}^\top \hat{\boldsymbol{z}}'$ for $b \in \{0, 1\}$. Each pair (u_b, v_b) is jointly Gaussian with unit variances and covariance ρ, so $\mathbb{E}[u_b v_b] = \rho$ and, by Isserlis' theorem, $\mathbb{E}[u_b^2 v_b^2] = \mathbb{E}[u_b^2]\mathbb{E}[v_b^2] + 2\,\mathbb{E}[u_b v_b]^2 = 1 + 2\rho^2$. The two projections are independent, so the component's contribution to the empirical kernel $K(\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}') = \phi(\hat{\boldsymbol{z}})^\top \phi(\hat{\boldsymbol{z}}')/m$ has mean

$$\mathbb{E}\big[(u_0 v_0)(u_1 v_1)\big] \;=\; \mathbb{E}[u_0 v_0]\,\mathbb{E}[u_1 v_1] \;=\; \rho^2,$$

and variance

$$\mathrm{Var}\big[(u_0 v_0)(u_1 v_1)\big] \;=\; \mathbb{E}[u_0^2 v_0^2]\,\mathbb{E}[u_1^2 v_1^2] - \rho^4 \;=\; (1 + 2\rho^2)^2 - \rho^4 \;=\; 1 + 4\rho^2 + 3\rho^4 \;\le\; 8,$$

since $|\rho| \le 1$. Components are independent across r, so averaging over the m components gives the kernel concentration

$$\begin{aligned} K(\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}') \;&=\; \frac{\phi(\hat{\boldsymbol{z}})^\top \phi(\hat{\boldsymbol{z}}')}{m} \;=\; \rho^2 + \eta, \qquad \mathbb{E}[\eta] = 0, \\ \mathrm{Var}\,\eta \;&=\; \frac{(1 + 2\rho^2)^2 - \rho^4}{m} \;=\; \frac{1 + 4\rho^2 + 3\rho^4}{m} \;\le\; \frac{8}{m}. \end{aligned} \tag{11}$$

Noise regime. Each centered summand $X_r = (u_0 v_0)(u_1 v_1) - \rho^2$ has tails heavier than any sub-Gaussian, but η averages m independent copies, so its small deviations are near Gaussian with the variance of Eq. 11. We use the sub-Gaussian form with variance proxy $\sigma^2 \le 8/m$ as a small-deviation approximation, and this step makes the analysis informal. A one-sided tail bound then gives

$$P(\eta > \Delta) \;\le\; \exp\big(-m\Delta^2/16\big),$$

for a margin Δ between the correct candidate's alignment and its nearest competitor's. A union bound over the N^2 ordered pairs of keys gives $P(\text{any failure}) \lesssim N^2 \exp(-m\Delta^2/16)$, so at fixed key geometry the failure probability decays exponentially in the width, and the width needed to hold it at a fixed level grows only logarithmically in the fact count. Rearranging $N^2 \exp(-m\Delta^2/16) \le 1$ gives the capacity law $m \ge 32 \ln N/\Delta^2$ that Section 4.4 tests against the measured frontier.

Geometry regime. When a cluster of keys is highly correlated, the margin is set not by noise but by the gap between the self-kernel, which concentrates at 1 with the same $8/m$ variance, and

the worst non-duplicate cross-kernel ρ^{*2}, with ρ^* the largest correlation among non-duplicate pairs (Section 4.4 estimates it by the 99.9th percentile of the pairwise correlations). Substituting $\Delta = 1 - \rho^{*2}$ into the same bound gives $P(\text{failure}) \lesssim N^2 \exp\big(-m(1-\rho^{*2})^2/16\big)$, where the margin enters squared in the exponent, so key geometry, not fact count, is the binding constraint. Because the bound is driven by the worst key pair alone, it is deliberately conservative as a numeric threshold, and Sections 4.3 and 4.4 report the comparison to measured key geometries rather than a point prediction.

The floor. When two edits share the subject and the entire prefix before the end of its mention, their keys are identical, since the key reads that left context alone (Section 3.2), and the argmax selection of Section 3.5 then degenerates to the earlier stored entry. Two entries that share a key $\hat{\boldsymbol{z}}$ but carry different values $\boldsymbol{y}, \boldsymbol{y}'$ produce identical feature vectors, so any readout that is linear in the features returns the same vector for both. In the idealized case of an isolated duplicated column, the ridge solution of Eq. 8 returns the mean $(\boldsymbol{y} + \boldsymbol{y}')/2$ at that key in the small-λ_r limit, since duplicating a column of $\boldsymbol{\Phi}$ splits its coefficient evenly under ridge. This failure is independent of m. No width distinguishes entries the feature map cannot tell apart.

Scope. The derivation above analyzes the projections at initialization. Refinement then fits the projections to their stored keys (Section 3.4), so the production matrices no longer follow Gaussian statistics. Their measured kernel falls about a hundredfold under the ρ^2 background where the pair mass concentrates and never returns to ρ^2 (Figure 4), so the capacity law is a conservative design guideline and the production memories exceed the capacity it certifies.

D SAME-WIDTH COMPARISON WITH A LINEAR READOUT

This appendix separates the capacity of the bilinear feature $\phi(\hat{\boldsymbol{z}}) = (\boldsymbol{A}_0\hat{\boldsymbol{z}}) \odot (\boldsymbol{A}_1\hat{\boldsymbol{z}})$ from the capacity of a linear readout at equal width. The linear arm projects each key to m dimensions with a Gaussian matrix $\boldsymbol{P} \in \mathbb{R}^{m\times d}$ whose entries are drawn from $\mathcal{N}(0, 1/d)$, re-normalizes each projection to unit norm, and fits its readout $\boldsymbol{W} \in \mathbb{R}^{d\times m}$ by the ridge regression of Eq. 8 with the same ridge 10^{-6}. The bilinear arm runs the construction of Sections 3.2–3.4 at hidden width m. Both arms store the same keys and the same centered value table. The pool consists of the production keys of GPT-J-6B on 10,000 CounterFact edits, the grid covers $m \in \{128, 256, 512, 1024, 2048\}$ and $N \in \{4000, 5000, 6000, 8000, 10000\}$, and each cell averages four seeds.

Table 3 reports readout recovery, the share of keys whose readout scores the stored target highest at every fact position over the full vocabulary. Recovery upper-bounds efficacy under the recall path of Section 3.5, where the gate snaps the fired position to the stored key before the readout.

The grid reads three ways. At the operating point $m = 2048$ and $N = 5{,}000$ the bilinear arm recovers 0.9951 of the keys against 0.9593 for the linear arm, so equal width leaves a 3.6-point gap where the fact count exceeds the width. The gap widens as width becomes scarce: at $m = 128$ the linear arm retains 0.1211 of the keys at $N = 4{,}000$ while the bilinear arm retains 0.8480, and at $N = 10{,}000$ the linear arm falls to 0.0784 against 0.6224. Width alone does not close the gap: at $N = 10{,}000$ the bilinear arm at $m = 1024$ recovers 0.9686 while the linear arm at the full key dimension $d = 4096$ recovers 0.9501, because the linear kernel passes the background correlation ρ at first order while the bilinear kernel suppresses it to ρ^2 (Appendix C).

Each grid cell is a single memory over N edits. The production system shards the edit stream at 5,000 edits, so every solve operates at $N = 5{,}000$, where the bilinear arm recovers 0.9951 of the keys. Appendix E reports the end-to-end comparison.

The gap is not specific to the CounterFact key pool. On the production keys of GPT-J-6B on 10,000 zsRE edits, the same measurement at the operating point gives 0.7197 for the linear arm against 0.9600 for the bilinear arm, a 24-point gap.

Table 4 takes the comparison end to end at the production operating point: six backbone–dataset cells at 10,000 edits with 5,000-edit shards, the linear arm swapping only the feature map for the Gaussian projection above. On CounterFact the end-to-end gap tracks the grid closely, 3.3 to 4.5 points of efficacy against the 3.6 points the grid predicts. On zsRE the gap largely closes, at most 1.3 points on LLaMA3-8B and none on the other two backbones, even though the grid gap on that key pool is larger. zsRE queries are often answerable by the backbone itself, which buffers a degraded

m	readout	N=4000	5000	6000	8000	10000
128	linear	0.1211	0.1057	0.0940	0.0846	0.0784
	bilinear	0.8480	0.7975	0.7506	0.6792	0.6224
256	linear	0.2512	0.1987	0.1695	0.1403	0.1233
	bilinear	0.9704	0.9434	0.9107	0.8614	0.8212
512	linear	0.5262	0.4236	0.3475	0.2587	0.2131
	bilinear	0.9933	0.9876	0.9780	0.9586	0.9377
1024	linear	0.8671	0.7767	0.6832	0.5306	0.4327
	bilinear	0.9954	0.9937	0.9892	0.9803	0.9686
2048	linear	0.9801	0.9593	0.9259	0.8455	0.7628
	bilinear	0.9966	0.9951	0.9922	0.9877	0.9819

Table 3: Readout recovery of the linear and bilinear readouts at equal width m, measured as the fraction of stored keys whose readout output places the largest logit on the stored target token over the full output table. Both arms store the same production keys of GPT-J-6B on CounterFact and the same centered value table, with four seeds per cell. Each cell is a single memory over N edits; the production system shards the edit stream so that every solve operates at $N = 5{,}000$ (Appendix E). The arms separate as width becomes scarce relative to the fact count.

		bilinear (production)			linear		
backbone	dataset	Eff	Gen	Spe	Eff	Gen	Spe
GPT-J-6B	CounterFact	98.8	45.6	10.9	95.5	44.4	11.0
	zsRE	98.9	98.0	25.8	98.7	97.9	25.8
LLaMA3-8B	CounterFact	98.2	80.1	18.5	94.8	78.0	18.5
	zsRE	96.5	96.0	38.0	95.2	94.8	38.0
Qwen3-4B	CounterFact	98.7	58.6	10.5	94.2	56.8	10.5
	zsRE	97.4	96.7	35.4	98.1	97.3	35.4

Table 4: End-to-end comparison of the bilinear and linear readouts at equal width $m = 2048$ under the production configuration (10,000 edits, 5,000-edit shards, frozen per-cell gate threshold). The linear arm differs only in the feature map, the Gaussian projection of Appendix D. On CounterFact the end-to-end gap tracks the grid prediction of Table 3 closely; on zsRE the gap largely closes end-to-end even though the grid gap on that key pool is larger.

readout, while CounterFact targets rare surface forms that leave no such fallback. The bilinear feature is therefore load-bearing at the production operating point, with the end-to-end signature strongest where the task offers no parametric fallback.

E SHARDING

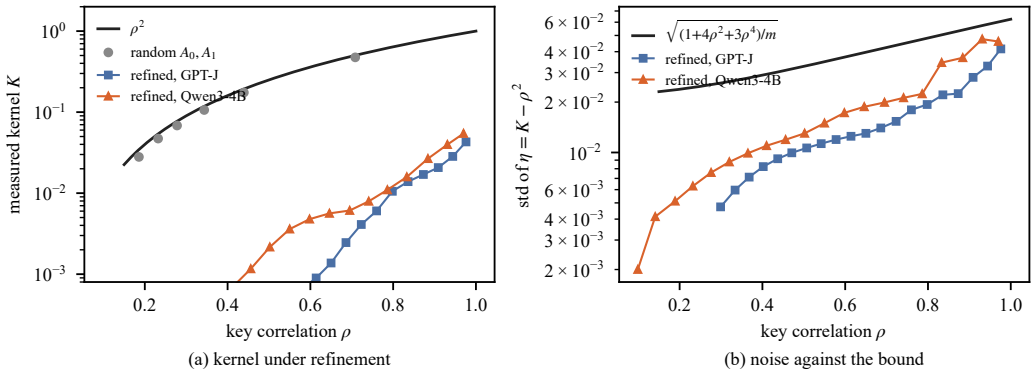
The production memory is built in consecutive shards of 5,000 edits (Appendix B). No parameters are shared across shards: the projections $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$ of each shard are refined on its own keys and its readout is solved against the same 5,000 keys. Every per-shard property therefore transfers to any stream length verbatim. This covers the refinement convergence of Figure 3(b) and the component ablations of Table 7.

Only the recall gate couples shards: it scores a token's key against the stored keys of every shard (Section 3.5). Specificity isolates exactly this element. It reads 10.3 at 5,000 edits against 10.9 at 10,000 edits on the same backbone and dataset (Table 1), so the gate is scale-flat within 0.6 points in the covered range.

Table 5 measures what sharding protects against. A single solve over the full stream degrades with N: efficacy drops by 0.5–1.5 points on CounterFact at 10,000 edits and by up to 7.7 points on zsRE. Generalization tracks the same slope with drops of up to 7.7 points on zsRE. The degradation follows the single-solve capacity curve of Appendix D. Specificity matches exactly in all twelve cells since

		single solve			sharded (production)		
backbone	dataset	Eff	Gen	Spe	Eff	Gen	Spe
GPT-J-6B	CounterFact	97.6	45.2	10.9	98.8	45.6	10.9
	zsRE	96.8	96.0	25.8	98.9	98.0	25.8
LLaMA3-8B	CounterFact	97.7	79.8	18.5	98.2	80.1	18.5
	zsRE	90.1	89.8	38.0	96.5	96.0	38.0
Qwen3-4B	CounterFact	97.2	58.2	10.5	98.7	58.6	10.5
	zsRE	89.7	89.0	35.4	97.4	96.7	35.4

Table 5: End-to-end effect of sharding at the operating width $m = 2048$ on 10,000-edit streams. The single-solve arm builds one memory over the full stream and the sharded arm is the production configuration of 5,000-edit shards. At 5,000 edits and below the two arms coincide by construction since one shard covers the stream. Both arms share the frozen per-cell gate threshold and the evaluation protocol of Table 1. Sharding preserves or improves efficacy and generalization in every cell and specificity matches across arms everywhere.



(a) kernel under refinement (b) noise against the bound

Figure 4: Measured kernel $K(\hat{\boldsymbol{z}}, \hat{\boldsymbol{z}}') = \phi(\hat{\boldsymbol{z}})^\top \phi(\hat{\boldsymbol{z}}')/m$ on within-shard key pairs of the production CounterFact memories at $m = 2048$. (a) Under Gaussian projections the kernel tracks ρ^2 as Figure 3(a) reports, while the refined production matrices hold a floor near 10^{-3} through the background and never rejoin ρ^2. (b) The per-bin spread of the deviation η against the informal bound.

the frozen gate threshold blocks neighborhood probes equally under one solve or two. Sharding therefore protects edit success rather than locality, which the gate already holds on its own.

The shard size follows the grid of Appendix D. At the operating width a single solve recovers 0.9951 of the keys at $N = 5{,}000$, holds 0.9922 at $N = 6{,}000$, and degrades to 0.9819 at $N = 10{,}000$. A depth of 5,000 keeps every solve inside the above-0.99 recovery region.

F KERNEL UNDER REFINEMENT

Figure 3(a) verifies the squared-similarity law under Gaussian projections. The alternating refinement of Section 3.4 adapts $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$ to the stored keys, so the production matrices leave the Gaussian regime. Figure 4 repeats the measurement with the production matrices of GPT-J-6B and Qwen3-4B-Instruct-2507 on their CounterFact pools. The refined kernel sits two orders of magnitude below ρ^2 across the background range $\rho \in [0.3, 0.65]$ and stays more than an order of magnitude below ρ^2 even inside the near-duplicate cluster above $\rho = 0.9$. Refinement thus suppresses cross-talk far beyond the squared law that random projections guarantee. Panel (b) tracks the deviation $\eta = K - \rho^2$: its per-bin spread stays under the informal bound of Appendix C across the full observed range.

		GRACE					HebbianEdit				
backbone	dataset	Eff	Gen	Spe	construction	inference	Eff	Gen	Spe	construction	inference
GPT-J-6B	CounterFact	99.7	0.5	14.2	2.38 s/edit (6.6 h)	+21.0%	98.8	45.6	10.9	2.7 min	+7.8%
	zsRE	99.7	31.1	27.0	2.38 s/edit (6.6 h)	+21.0%	98.9	98.0	25.8	5.3 min	+7.8%
LLaMA3-8B	CounterFact	99.9	0.4	21.4	2.36 s/edit (6.6 h)	+11.3%	98.2	80.1	18.5	2.6 min	+4.6%
	zsRE	99.9	40.5	38.4	2.36 s/edit (6.6 h)	+11.3%	96.5	96.0	38.0	14.4 min	+4.6%
Qwen3-4B	CounterFact	99.3	0.4	13.8	3.97 s/edit (11.0 h)	+7.6%	98.7	58.6	10.5	1.8 min	+5.1%
	zsRE	99.6	36.7	35.7	3.97 s/edit (11.0 h)	+7.6%	97.4	96.7	35.4	3.6 min	+5.1%

Table 6: Performance and wall-clock cost at 10,000 edits on one RTX 4090, per backbone and dataset. Eff, Gen, and Spe follow the main-table protocol at $T = 10{,}000$. Construction is the write cost of the full edit stream: GRACE trains each edit online (2.36–3.97 seconds per edit, 6.6–11.0 hours per stream), while the memory is built once in 1.8–14.4 minutes. Inference is the per-forward overhead at the evaluation batch shape, each measured against the bare model in the same session.

G Construction and inference cost

Table 6 times both stages against GRACE (Hartvigsen et al., 2023) on the same machine, the same backbones, and the same 10,000 edits per dataset. GRACE writes each edit online with 50 forward-backward steps at 2.36 to 3.97 seconds per edit across the three backbones. The full stream costs 6.6 to 11.0 hours, while the memory is built once in 1.8 to 14.4 minutes. At inference, the GRACE adapter scores each forward against its codebook and adds 7.6% to 21.0% per forward at the evaluation batch shape. The recall path adds a scoring pass over the stored keys and a fixed-width FactMLP evaluation per forward, costing 4.6% to 7.8% under the same measurement. The performance columns follow the main-table protocol at this scale: efficacy is saturated on both sides. Specificity favors GRACE. On CounterFact it stays within 0.4 points of the pre-edited model where the memory drops 3.3 to 3.5, and on zsRE both sides stay within 1.2 points of it. The lookup gate opens mostly on verbatim repeats (Gen at most 0.5 on CounterFact and 40.5 on zsRE) while the recall path generalizes across phrasings (Gen 45.6 to 80.1 on CounterFact and 96.0 to 98.0 on zsRE).

H Ablation study

Ablations run on GPT-J-6B with 5,000 CounterFact edits at gate threshold $\tau_q = 0.90$. The full method scores $0.990/0.872/0.095$ (efficacy/generality/specificity), and every variant is evaluated on the same items at the same threshold. The injection-site sweep of Table 8 runs on Qwen3-4B-Instruct-2507 with 50 CounterFact edits.

Table 7 reports every variant. Two components are load-bearing: removing the alternating refinement collapses the readout to zero, and sourcing keys from the embedding table collapses it likewise. The shallow layer combination is the joint source of recognition and locality: layer 0 alone loses specificity, single layers 1–3 decay in generality, and a deep-layer combination drops generality to 0.415.

Injection site and strength. Table 8 sweeps α from 8 to 512 at the production site and at the midpoint layer. The production site attains full efficacy from $\alpha = 32$ upward, while the mid-layer site peaks at 24/50 and degrades at higher strengths.

variant	Eff	Gen	Spe
full method	0.990	0.872	0.095
readout construction			
random features, whitened keys	0.000	0.000	0.095
random features, raw keys	0.000	0.000	0.093
key source			
keys from word embeddings	0.004	0.005	0.144
layer combination			
layer 0 only	0.988	0.727	0.024
layer 1 only	0.990	0.817	0.092
layer 2 only	0.990	0.726	0.113
layer 3 only	0.990	0.659	0.118
deep layers	0.988	0.415	0.123

Table 7: Ablation on GPT-J-6B with 5,000 CounterFact edits at gate threshold $\tau_q = 0.90$. Each row replaces one component of the full method and is evaluated on the same items at the same threshold. The alternating refinement, the hidden-state key source, and the shallow layer combination are load-bearing.

α	8	32	64	128	256	512
Mid-layer (layer 18)	1	6	22	24	11	10
After final norm (ours)	10	50	50	50	50	50

Table 8: Efficacy (hits out of 50 CounterFact edits, Qwen3-4B-Instruct-2507) when the memory value is injected at the midpoint layer versus the production site of Eq. 3, across injection strengths α.