# HASH-EMBEDDING MEMORY INJECTION: CROSS-SCALE TRANSFERABLE DIFFERENTIAL MEMORY FOR LARGE LANGUAGE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Fine-tuning is expensive and adapters such as LoRA are tied to one base model, so domain knowledge is voided at every backbone upgrade. We propose Hash-Embedding Memory Injection (HEMI), which stores knowledge in a hash $n$-gram table trained once on a source backbone and reused across backbones of the same family and pre-training lineage through each backbone's anchor basis. Three contributions: (1) a critical-frequency theory giving a closed-form $f^*$ and the table-size bound $V^* = T \cdot L/f^*$; pruning at $f^*$ costs under $1\%$ held-out loss over $48$ configurations in 3 domains; (2) differential injection of only the residual between stored memory and the hidden state, under evidential gating and RMS balancing—ablations show the gate and the adapter decide whether a transferred table helps or hurts (removing the adapter hurts in all three settings; removing the gate does so in two and leaves $\sim 25\%$ of the gain in the third), while dropping differencing or RMS costs $0.005$–$0.039$ nats; (3) anchor rendering via thin QR factorization, plus a migration stability theorem bounding direction error by three additive sources: finite-frequency noise, Stage-2 drift, and anchor-subspace mismatch. On DeepSeek-R1-Distill-Qwen, a table trained on 1.5B improves the 7B target in four domains by $+0.150$ to $+0.284$ nats with $65{,}811$ target-side parameters ($1/307$ of LoRA at $r = 8$), on top of a 7.68M-parameter table trained once on the source; the gain does not survive on the 14B target ($-0.003$ on PubMed, $-0.149$ on Math; $+0.071 \pm 0.043$ against $+0.238 \pm 0.006$ on 7B under a second protocol), where a zero-table control gains more than the trained table ($+0.127$ on PubMed, $+0.045$ on Math), so the content of the table contributes nothing there.

## 1 INTRODUCTION

An enterprise agent for a vertical domain accumulates private knowledge through incremental training on a base model (product terminology, business workflows, compliance rules). When a stronger base model ships and the business upgrades, that memory does not carry over. Full fine-tuning writes it into the weights and must be rerun on the new model at essentially the same cost. PEFT methods such as LoRA and Adapters (Hu et al., 2022; Houlsby et al., 2019) tie their updates to the original weight matrices and cannot be carried over as-is. RAG (Lewis et al., 2020) avoids training but keeps knowledge in an external document buffer, paying context-window tokens at every inference call. In all three cases domain knowledge ends up either in backbone-specific weights or outside the model, so upgrading the base model means rebuilding it from scratch. This paper studies a narrow version of the problem: can a memory table trained on a small member of a model family be carried over to a larger one with minimal additional training, within the same family and pre-training lineage?

Engram (Cheng et al., 2026) is the closest existing attempt: it stores memory as direction vectors in an $n$-gram hash table, retrieves one vector per token by hashing the $n$-gram, and adds it to the hidden state of a designated layer, introducing knowledge without consuming context or training a large adapter. Three design choices tie the table to the model it was trained on. First, the projection from memory space to hidden space is a trained matrix whose shape depends on the source backbone's hidden size, so the table cannot be mounted on a backbone of a different width. Second, the memory vector is added wholesale regardless of what the target already represents, so a stronger backbone



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.

receives content it may already have. Third, $V$ is chosen by search with no principled rule: too large and most slots are never trained, too small and unrelated $n$-grams collide.

We propose Hash-Embedding Memory Injection (HEMI), which addresses these three limitations directly.

- **Anchor rendering.** Memory directions live in a backbone-independent space and are rendered into a target backbone's hidden space via an orthonormal basis computed from the backbone's own anchor-token embeddings, with no training required. Mounting a table on a new backbone changes neither the backbone nor the table; only a small alignment pack (a scale vector, a gate and a low-rank adapter) is trained on the target. A migration stability bound relates rendering error to training frequency, injection magnitude and anchor-subspace mismatch.
- **Differential injection.** Instead of adding the memory vector wholesale, HEMI projects the hidden state into memory space, subtracts it from the retrieved vector and injects only the residual. Content the backbone already represents cancels out, and an evidential gate suppresses injection where memory and model agree.
- **Critical-frequency pruning.** A signal-to-noise analysis of slot updates yields a closed-form critical frequency $f^*$ below which a slot's memory is indistinguishable from gradient noise, giving a principled table-size bound $V^* = T \cdot L/f^*$ and a pruning rule that removes under-trained slots after training.

We evaluate HEMI on DeepSeek-R1-Distill-Qwen at 1.5B, 7B and 14B across PubMed, Math, Code and General web text. A table trained on 1.5B and mounted on 7B lowers validation loss on all four domains by $0.150$–$0.284$ nats with 65,811 target-side parameters, about $1/307$ of LoRA at rank 8. Ablations identify the gate and the adapter as the components that decide whether a transferred table helps or hurts (Section 4.4), and pruning at $f^*$ costs under $1\%$ held-out loss against an oracle threshold over $48$ configurations. The gain does not survive on the 14B target, where an empty table gains more than the trained one (Section 4.5). All experiments stay within one model family and one pre-training lineage; cross-family transfer is future work.

## 2 Related Work

**Memory-augmented language models.** Retrieval-augmented generation (Lewis et al., 2020), RETRO (Borgeaud et al., 2022) and kNN-LM (Khandelwal et al., 2020) keep knowledge outside the model. A second line stores it inside: product-key memory (Lample et al., 2019) attaches key-value tables to the hidden state and Memory Decoder (Cao et al., 2025) plugs a separate memory model into a backbone. Engram (Cheng et al., 2026) replaces learned keys with $n$-gram hashing, making retrieval deterministic and training-free. Two lines target reuse across models: TransMem (Lei et al., 2026) transfers hidden-state memory and Li et al. (2026) transfer it through a target-side reader; both move a representation tied to the source's hidden geometry, whereas HEMI moves a hashed table of directions in a backbone-independent space, read through each target's own anchor basis. HEMI keeps Engram's table and retrieval and changes what follows a fetch: projection (anchor rendering), combination with the hidden state (differential injection), and size (critical-frequency pruning).

**Parameter-efficient fine-tuning.** LoRA (Hu et al., 2022), Adapters (Houlsby et al., 2019) and prefix tuning (Li & Liang, 2021) freeze the backbone and train few parameters attached to its weights; defined relative to one backbone, they need a transfer step per upgrade (Wang et al., 2024; Farhadzadeh et al., 2025) and cannot carry knowledge learned on another backbone to the target. HEMI trains comparably few parameters—a scale vector, a gate, an adapter—but retrains them per target; what is reused is the table, whose directions live in the backbone-independent space $\mathbb{R}^{d_m}$. ReFT (Wu et al., 2024) is closest in form, adding back a difference $h + R^\top(Wh + b - Rh)$; HEMI replaces the learned target $Wh + b$ by a hash-addressed table entry from another backbone and fixes the subspace by anchor tokens (Appendix D.1).

**Domain knowledge injection.** Continued pre-training on domain corpora (Gururangan et al., 2020) is the standard alternative and must be rerun on every backbone. That fine-tuning absorbs un-

seen knowledge only weakly (Gekhman et al., 2024; Biderman et al., 2024) is the premise HEMI's table addresses.

**Cross-model representation alignment.** Relative representations (Moschella et al., 2023) express an embedding through its similarities to anchor samples, making independently trained latent spaces comparable. Anchor rendering uses the same intuition with anchor tokens, rendering memory through an orthonormal basis of their embeddings so that a direction is mapped isometrically into each backbone separately; the subspaces on different backbones stay offset by the principal angles of Section 3.2, which enter Theorem 1 through $\delta_{\max}$. Model stitching (Bansal et al., 2021) learns a linear map between networks instead; distillation (Hinton et al., 2015) and weight interpolation (Matena & Raffel, 2022) yield a new model, not a shared memory.

**Knowledge editing.** ROME (Meng et al., 2022), MEMIT (Meng et al., 2023) and MEMOIR (Wang et al., 2025) edit single facts in the weights, precisely but a few at a time. Differential injection is a softer, batched alternative: each slot carries a cluster of $n$-gram examples and a layer, and the gate decides whether to act. Catastrophic forgetting (Kirkpatrick et al., 2017) protects knowledge under further training; HEMI protects it when the model is replaced.

# 3 METHOD

HEMI has three stages. Stage 0 builds the anchor set $\mathcal{S}$ and the rendering matrix $\Phi_B$ by thin QR on the backbone's own embedding matrix (Section 3.2); Stage 1 trains an $n$-gram memory table on a source backbone and prunes it at the critical frequency (Section 3.4); Stage 2 renders the frozen table into a target backbone and injects it as a gated residual while training only a small alignment pack (Section 3.3). Section 3.5 bounds the direction error of the rendered memory across backbones.

Table 3 in Appendix B.1 lists the notation.

## 3.1 PRELIMINARIES: THE ENGRAM MEMORY TABLE

Engram (Cheng et al., 2026) attaches a table $M \in \mathbb{R}^{V \times d_m}$ to one layer; the trailing $n$-gram at position $i$ is mapped to a slot by

$$\mathrm{slot}(g_i) = \left(\sum_{j=0}^{n-1} x_{i-n+1+j}\, p^j\right) \bmod V, \tag{1}$$

with fixed prime $p = 100003$, and $v_k = M[\mathrm{slot}(g_i)]$ is retrieved and trained by backpropagation (colliding $n$-grams share one row). HEMI keeps the hashing and the table and replaces three components: the backbone-specific projection (§3.2), the wholesale additive injection (§3.3), and the unprincipled table size (§3.4).

## 3.2 ANCHOR RENDERING AND THE TRANSFER PROTOCOL

**Rendering matrix.** Let $E_B \in \mathbb{R}^{|\mathrm{vocab}| \times d_h}$ be the input embedding matrix of backbone $B$ and $\mathcal{S}$ be a fixed set of $K$ anchor tokens shared by all backbones under consideration. HEMI takes the anchor rows $E_B[\mathcal{S}] \in \mathbb{R}^{K \times d_h}$ and computes a thin QR factorization of their transpose,

$$E_B[S]^\top = QR, \qquad \Phi_B = Q[:, : d_m] \in \mathbb{R}^{d_h \times d_m}. \tag{2}$$

$\Phi_B$ has orthonormal columns, so $\langle \Phi_B u, \Phi_B v \rangle = u^\top v$ for all $u, v \in \mathbb{R}^{d_m}$: a memory direction is rendered into $B$'s hidden space without changing its length or its angles to other directions. $\Phi_B$ is computed from $B$'s weights, receives no gradient, and is the only backbone-specific object needed to render the table; the alignment pack $\theta$ (Table 3) is the only one that is trained. Memory directions live in $\mathbb{R}^{d_m}$ and are shared: the same $v_k$ is rendered as $\Phi_S v_k$ on the source and $\Phi_B v_k$ on the target. The two renderings need not agree: Theorem 2 bounds their disagreement by the largest principal angle $\theta_{\max}$ between the anchor subspaces, which enters Theorem 1 as $\delta_{\max} = 2\sin(\theta_{\max}/2)$. We do not measure $\theta_{\max}$ on real backbones. What we measure is the end-to-end quantity the theorem bounds, the direction error $1 - \cos(p_S, \Pi p_B)$: $0.048$ at $K = d_m = 128$ and $0.046$ at $K = 256$ for the $1.5\mathrm{B}\rightarrow$ 7B pair, rising to $0.52$ at $K = 16$ where the rank condition fails (Appendix C.3.2); no such measurement is available for the $1.5\mathrm{B}\rightarrow$ 14B pair. That error aggregates all three sources

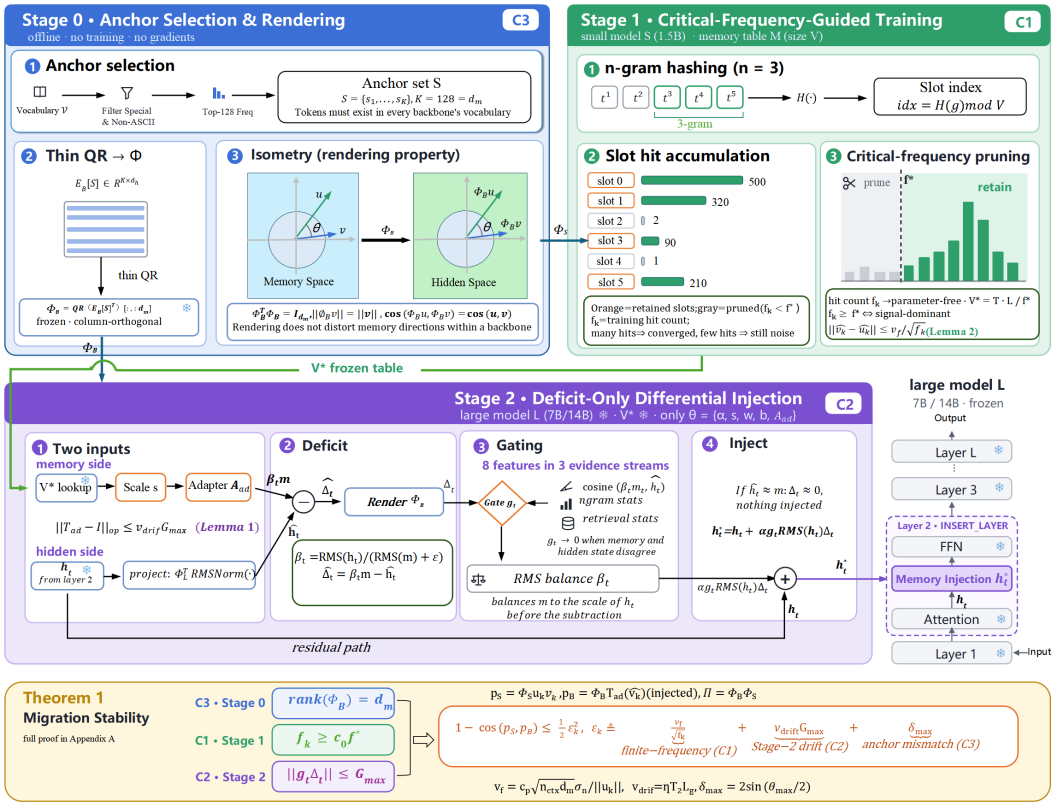


Figure 1: Overview of HEMI. Stage 0: anchor set $S$ and rendering matrix $\Phi_B$ by thin QR (§3.2). Stage 1: train the hash $n$-gram table $M$ on a small backbone and prune at $f^*$ (§3.4). Stage 2: render the frozen table via $\Phi_B$ and inject the gated residual $\beta m - \hat{h}$ (§3.3). Only $\theta$ is trained on the target.

of Theorem 1 and cannot be attributed to the anchor term alone; the $\delta_{\max}^2/2$ scaling is checked separately on synthetic constructions (Appendix C.6, V3: fitted exponent 2.00 against a theoretical 2). The QR convention (sign of $R$'s diagonal) is fixed for reproducibility.

**Anchor set.** $S$ must exist in every backbone's vocabulary and satisfy $\mathrm{rank}(\Phi_B) = d_m$ with $K \geq d_m$; we use the $K = d_m$ highest-frequency tokens of a mixed-domain corpus (alternatives and a sweep over $K$: Appendix B.3).

**Three-stage protocol.** **Stage 0.** Select the anchor set $S$ shared by all backbones in the family and compute $\Phi_S$ and $\Phi_B$ by thin QR on each backbone's embedding matrix; this step is offline and training-free. **Stage 1 (source $S$).** Train the table $M$ and the alignment pack $\theta$ on domain data with the injection rule of Section 3.3, then prune $M$ at $f^*$ (Section 3.4); the pruned table is the transferable asset. **Stage 2 (target $B$).** Load the frozen table and $\Phi_B$ and train only $\theta$ on target-domain data; $B$'s weights and $M$ are not updated. Every number reported as HEMI in Section 4 uses this protocol, with one deviation noted there: a table trained from scratch on the target.

### 3.3 Differential Injection

At the injection layer, let $h \in \mathbb{R}^{d_h}$ be the hidden state at one token and $v_k$ the retrieved memory direction. HEMI first expresses $h$ in memory space and the memory in the target's coordinates,

$$\hat{h} = \Phi_B^\top \mathrm{RMSNorm}(h) \in \mathbb{R}^{d_m}, \qquad m = A_{\mathrm{ad}}(s \odot v_k) \in \mathbb{R}^{d_m}, \tag{3}$$

where $s \in \mathbb{R}^{d_m}$ is a per-dimension scale and $A_{\mathrm{ad}} \in \mathbb{R}^{d_m \times d_m}$ is an identity-initialized adapter that absorbs the residual misalignment between the source and target anchor subspaces. A

freshly trained table has much smaller norm than $h$, so $m$ is rescaled by the balance factor $\beta = \mathrm{RMS}(h)/(\mathrm{RMS}(m) + \epsilon)$. The injected quantity is the residual between them,

$$\Delta = \beta m - \hat{h}, \qquad h \leftarrow h + \alpha\, g\, \mathrm{RMS}(h)\, \Phi_B \Delta, \tag{4}$$

with a sigmoid-parameterized magnitude $\alpha \in [0, 1]$ and a gate $g \in [0, 1]$ (Table 3). When the backbone already encodes what the slot stores, $\beta m \approx \hat{h}$ and the injection vanishes; when it does not, $\Delta$ supplies the missing component. Direct injection, $h \leftarrow h + \alpha\beta\, \Phi_B v_k$, adds the full vector regardless of $h$ and is the baseline for the residual-training ablations of Appendix C.1.3.

**Evidential gate.** The gate is a logistic function of eight features, $g = \sigma(w^\top e + b) \cdot \mathrm{mask}$ ($\mathrm{mask}$ zeros padding): three token-level features (memory–hidden cosine, running hit rate, hash-dispersion bit count), four Stage-1 slot statistics, and the per-token miss rate; the full list and the role of each feature are in Appendix B.5. $w \in \mathbb{R}^8$ and $b$ belong to $\theta$.

### 3.4 Critical-frequency pruning

A slot is updated only when an $n$-gram hashes to it, and with $T$ training samples of $L - n$ positions each the expected hit count of a slot is $\mathbb{E}[f_k] = T \cdot L/V$. Each hit adds to the slot a coherent signal $u_k$ along with the gradient noise of the $n_{\mathrm{ctx}}$ context positions that share the slot at that step. After $f_k$ hits the signal grows linearly in $f_k$ while the noise grows as a random walk, so where gradient noise dominates the initialization noise the signal-to-noise ratio is

$$\mathrm{SNR}(f_k) \approx \frac{f_k \|u_k\|^2}{n_{\mathrm{ctx}} d_m\, \sigma_n^2}, \tag{5}$$

with $\sigma_n^2$ the per-dimension gradient-noise variance. Requiring a typical ratio $\mathrm{SNR}_0$, set by the recall and false-positive rates of a binary detection test, gives the critical frequency

$$f^* = \mathrm{SNR}_0\, \frac{n_{\mathrm{ctx}} d_m\, \sigma_n^2}{\|u_k\|^2}, \tag{6}$$

below which the slot's content cannot be distinguished from noise (Appendix B.2). Under (A1) $\|u_k\|^2 \approx d_m \sigma_n^2$ and (A2) $\mathrm{SNR}_0 = 1$, $f^* \approx n_{\mathrm{ctx}}$, the value used in all experiments; both assumptions and their relaxation are in Appendix B.2. Two consequences follow. Requiring a typical slot to reach $f^*$ gives the table-size bound $V^* = T \cdot L/f^*$: larger tables leave most slots below threshold, smaller tables force unrelated $n$-grams to share slots. After training, slots with $f_k < f^*$ can be removed by zeroing their rows; the instantiation ($V^* \approx 6 \cdot 10^4$) and its validation are in Section 4 and Appendix C.2.

### 3.5 Migration stability

The three components interact: a slot must be trained often enough for its direction to be meaningful (Section 3.4), the target rendering must preserve that direction (Section 3.2), and Stage-2 training must not drift it (Section 3.3). The following bound makes the interaction explicit.

**Lemma 1 (adapter drift)** *Let $T_{\mathrm{ad}}(x) = A_{\mathrm{ad}}(s \odot x)$ be the alignment-pack map, initialised at $s = \mathbf{1}$, $A_{\mathrm{ad}} = I$ so that $T_{\mathrm{ad}} = I$, and updated by SGD with step size $\eta$ for $T_2$ steps in Stage 2. Assume (a) $\|g_t \Delta_t\| \le G_{\max}$ for every $t$, and (b) $\|\partial \mathcal{L}/\partial T_{\mathrm{ad}}\|_F \le L_g \|g_t \Delta_t\|$ for a constant $L_g$ bounding the upstream gradient. Then, with $\nu_{\mathrm{drift}} \triangleq 2\eta T_2 L_g$, for every unit vector $x$*

$$\left\| \overline{T_{\mathrm{ad}} x} - x \right\| \le 2 \|T_{\mathrm{ad}} - I\|_{\mathrm{op}} \le 2 \|T_{\mathrm{ad}} - I\|_F \le 2\eta T_2 L_g\, G_{\max} = \nu_{\mathrm{drift}}\, G_{\max}, \tag{7}$$

*the factor 2 being the Lipschitz constant of normalisation (Appendix A).*

**Lemma 2 (finite-frequency direction error)** *Under the slot-update model of Section 3.4, let $u_k$ be the coherent signal accumulated by slot $k$ and $v_k$ its stored content after $f_k$ hits, and let $\hat{u}_k = u_k/\|u_k\|$, $\hat{v}_k = v_k/\|v_k\|$. Then with probability at least $1 - \rho$, for a constant $c_\rho$ depending only on $\rho$,*

$$\|\hat{v}_k - \hat{u}_k\| \le \frac{\nu_f}{\sqrt{f_k}}, \qquad \nu_f \triangleq c_\rho\, \frac{\sqrt{n_{\mathrm{ctx}} d_m}\, \sigma_n}{\|u_k\|}, \tag{8}$$

*which reduces to $\nu_f = c_\rho \sqrt{n_{\mathrm{ctx}}}$ under (A1).*

**Theorem 1 (migration stability)** *Let $u_k$ be the coherent signal accumulated by slot $k$ in Stage 1 and $\hat{u}_k = u_k/\|u_k\|$; let $v_k$ be the content actually stored, with $\hat{v}_k = v_k/\|v_k\|$ (Lemma 2), and let $T_{\mathrm{ad}}$ be the alignment-pack map at the end of Stage 2 (Lemma 1). Define the* intended *direction $p_S = \Phi_S \hat{u}_k$ and the* injected *direction $p_B = \Phi_B T_{\mathrm{ad}}(\hat{v}_k)$. Suppose (i) $\Phi_S$ and $\Phi_B$ have orthonormal columns in a common ambient space $\mathbb{R}^D$, $D \geq \max(d_h^S, d_h^B)$, obtained by identifying the two hidden spaces through their anchor-token correspondence; the ambient space is* not *a block-diagonal product, for which $\theta_{\max} = 90°$); let $\Pi = \Phi_B \Phi_B^\top$ denote the orthogonal projection onto $\mathrm{range}(\Phi_B)$, so that $\Pi p_B = p_B$ since $p_B \in \mathrm{range}(\Phi_B)$; (ii) $\mathrm{rank}(\Phi_B) = d_m$; (iii) $f_k \geq c_0 f^*$ for some constant $c_0 \geq 1$; and (iv) the gated injection satisfies $\|g_t \Delta_t\| \leq G_{\max}$ throughout Stage 2. Then*

$$1 - \cos(p_S, p_B) \leq \tfrac{1}{2}\varepsilon_k^2, \varepsilon_k \qquad \triangleq \qquad \underbrace{\frac{\nu_f}{\sqrt{f_k}}}_{\text{finite-frequency (C1)}} + \underbrace{\nu_{\mathrm{drift}}\, G_{\max}}_{\text{Stage-2 drift (C2)}} + \underbrace{\delta_{\max}}_{\text{anchor mismatch (C3)}}, \tag{9}$$

*where $\delta_{\max} = 2\sin(\theta_{\max}/2)$ measures the mismatch between the two anchor subspaces, $\nu_f$ is the finite-frequency constant of Lemma 2, and $\nu_{\mathrm{drift}}$ the Stage-2 drift constant of Lemma 1. Appendix A bounds the total angle between the intended and the injected direction by $\varepsilon_k$ and applies $1 - \cos \leq \frac{1}{2}\|\cdot\|^2$.*

The bound has a single term: the total error angle $\varepsilon_k$ sums the slot's finite-frequency noise (Lemma 2, C1), the Stage-2 drift of the alignment pack (Lemma 1, C2) and the anchor-subspace mismatch (C3), so any of the three can dominate on its own. Since $\Pi p_B = p_B$, the bound measures the direction agreement between the intended rendering $p_S$ and the injected rendering $p_B$ directly. Because $p_S$ is defined against the slot's signal direction $\hat{u}_k$ rather than the stored $\hat{v}_k$, finite-frequency noise enters the bound instead of cancelling between the two renderings.The Cauchy–Schwarz form and cross-terms are in Appendix A; tightness and each condition's role are in Appendix A.1, and the predicted scaling is checked numerically in Appendix C.6.

## 4 Experiments

The experiments ask whether a transferred table helps (Section 4.2), what is transferred (Section 4.3), which parts of the injection rule matter (Section 4.4), and where transfer stops (Section 4.5).

### 4.1 Setup

**Backbones.** DeepSeek-R1-Distill-Qwen 1.5B ($d_h = 1536$), 7B ($d_h = 3584$) and 14B ($d_h = 5120$), which share a tokenizer and therefore the anchor token ids. The 1.5B and 7B models are distilled from Qwen2.5-Math checkpoints and the 14B model from a general Qwen2.5 checkpoint. Unless stated otherwise, tables and reusable baselines are trained on 1.5B and evaluated on 7B.

**Domains and data.** PubMed (biomedical abstracts), Math (mathematical web text), Code (Python source) and General (web text), with 12,000 training samples per domain (5,000 for Code) and 500 validation samples from a fixed 80/20 temporal split (Appendix E.1). The first 10% of the validation set is a development part used to choose every interpolation weight and learning rate; all numbers are on the remaining 90%.

**Metric.** Validation loss in nats on the test part, reported as the gain $\Delta L = L_{\mathrm{Base}} - L_{\mathrm{method}}$ over the unmodified backbone, so positive is better. Entries are from seed 42 unless the caption states the seeds. Configurations differ between sections (the main table uses $V = 60$k and 12k samples) and numbers are compared only within a configuration.

**HEMI configuration.** $V = 60{,}000$, $d_m = 128$, $n = 3$, injection at layer 2, $K = 128$ anchors chosen by corpus frequency (Appendix B.3), $f^* = n_{\mathrm{ctx}}$; $\theta$ has 65,811 parameters. Optimizer, learning rates and training length are in Appendix E.3.

**Baselines.** Three groups, with implementations and the ranges searched on the development part in Appendix E.2: *non-parametric* (in-context RAG (Ram et al., 2023)); *target-side fine-tuning* (LoRA (Hu et al., 2022) at $r = 64$, 8 and 1, and prompt tuning (Lester et al., 2021), the last two at the budget of $\theta$); and *trained once on 1.5B and reused* (Kneser-Ney 3-gram, Memory Decoder (Cao

Table 1: Gain in validation loss over Base on the 7B backbone; the Base row gives its loss. "Target params" counts parameters trained on 7B; "Reusable" marks components trained on 1.5B and carried over without retraining ("table": only the alignment pack is retrained); "Extra fwd" is extra forward passes per token at inference. Seed 42; R: a retrieval step.

| Method | Target params | Reusable | Extra fwd | PubMed | Math | Code | General |
|---|---|---|---|---|---|---|---|
| Base | 0 | – | 0 | 2.868 | 2.335 | 1.946 | 3.702 |
| *Non-parametric, no training on the target* | | | | | | | |
| RAG (BM25, in-context) | 0 | – | R | +0.013 | +0.008 | +0.028 | +0.019 |
| *Fine-tuning on the target* | | | | | | | |
| LoRA $r$=64, all projections | 161.5M | no | 0 | +0.345 | +0.312 | +0.258 | +0.403 |
| LoRA $r$=8 | 20.2M | no | 0 | +0.340 | +0.305 | +0.239 | +0.397 |
| LoRA $r$=1 | 201K | no | 0 | +0.282 | +0.237 | +0.193 | +0.333 |
| Prompt tuning | 65K | no | 0 | +0.244 | +0.192 | +0.170 | +0.299 |
| *Trained once on the 1.5B model, reused on 7B* | | | | | | | |
| Kneser-Ney 3-gram, output interpolation | 0 | yes | 0 | +0.062 | +0.038 | +0.039 | +0.070 |
| Memory Decoder | 0 | yes | 1 | +0.224 | +0.167 | +0.151 | +0.242 |
| Proxy-tuning | 0 | yes | 2 | +0.128 | +0.129 | +0.074 | +0.173 |
| CombLM | 0 | yes | 1 | +0.287 | +0.242 | +0.200 | +0.342 |
| **HEMI** | 65.8K | table | 0 | +0.247 | +0.150 | +0.182 | +0.284 |
| *Engram injection rule, reduced protocol of Appendix C.5; not comparable with the rows above* | | | | | | | |
| Engram, frozen 1.5B table, projection retrained on 7B | 26.2M | no | 0 | −0.254 | −0.440 | −0.441 | – |
| HEMI, same reduced protocol | 65.8K | table | 0 | +0.172 | +0.082 | +0.095 | – |
| *Combination* | | | | | | | |
| HEMI + LoRA $r$=8, joint | 20.3M | – | 0 | +0.380 | +0.463 | +0.496 | +0.374 |
| HEMI then LoRA $r$=8 | 20.2M | – | 0 | +0.398 | +0.511 | +0.295 | +0.377 |

et al., 2025), proxy-tuning (Liu et al., 2024; Mitchell et al., 2024), CombLM (Ormazabal et al., 2023)). All share the data, split and evaluation code.

### 4.2 Main Comparison

**Against target-side fine-tuning.** HEMI lowers the loss in all four domains by 0.150–0.284 nats with 65,811 target-side parameters; LoRA lowers it further in every domain (mean 0.320 at $r = 8$ and 0.330 at $r = 64$, against 0.216), so HEMI does not replace fine-tuning when it is affordable. At matched budget LoRA at $r = 1$, with three times the parameters of $\theta$, gains more in every domain (0.282, 0.237, 0.193, 0.333 against 0.247, 0.150, 0.182, 0.284) but only a third to a half as much per parameter; prompt tuning at the budget of $\theta$ is within a few hundredths of HEMI in all four. In-context retrieval changes the loss by at most 0.03.

**Against other reusable methods.** Every method in the third group trains its domain component once on 1.5B and reuses it on 7B. A Kneser-Ney 3-gram interpolated at the output gains 0.038–0.070, at most a quarter of HEMI, so $n$-gram statistics alone do not account for the table. Proxy-tuning, which reuses an adapted 1.5B model at two extra forward passes per token, is below HEMI in all four. Memory Decoder is ahead on Math by 0.017 and behind on the other three by 0.023–0.042; CombLM is ahead in all four by 0.018–0.092. Both train nothing on the target but run the 1.5B model at every token, whereas HEMI adds only a table lookup: the reusable methods that gain more pay for it with a second model at inference.

**Against the Engram injection rule.** Appendix C.5 implements the rule HEMI replaces—retrieve one vector per $n$-gram and add it to the hidden state—under a budget shared by every variant (1k samples, 2 epochs, $V$ = 20k, layer 10). It gives no gain on any backbone, whether the frozen table's projection is retrained on the target with 26.2M parameters or the table is retrained from scratch on 7B with 28.7M, while HEMI under the same budget, with the same frozen table, gains +0.172, +0.082 and +0.095 (Table 1). The Engram variants train about 400× more target-side parameters than HEMI's alignment pack and still fail, so the difference is in the injection rule and not in capacity. What these runs do not show is that transfer is what breaks the rule: it gives no gain natively either, on the backbone the table was trained on.

Table 2: Attribution of HEMI's gain to the content of the table, on the 1.5B backbone with $V = 200$k and 1k samples; mean $\pm$ std over three seeds.

| Table content | PubMed | Math |
|---|---|---|
| zero (content $\approx 0$) | $+0.138 \pm 0.005$ | $+0.087 \pm 0.004$ |
| random (no domain knowledge) | $+0.201 \pm 0.016$ | $+0.102 \pm 0.029$ |
| trained, rows shuffled | $+0.221 \pm 0.017$ | $+0.128 \pm 0.033$ |
| trained (HEMI) | $+0.339 \pm 0.005$ | $+0.071 \pm 0.004$ |

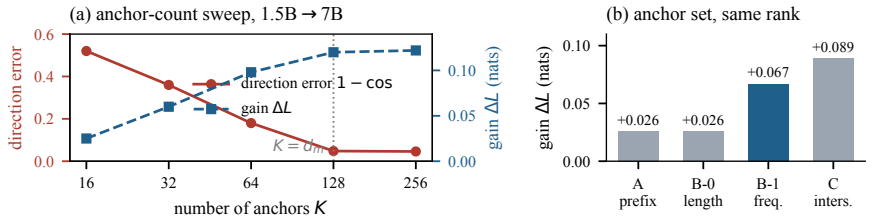


Figure 2: Anchor rendering on the 1.5B→7B transfer. (a) Direction error and gain vs. anchor count $K$; both saturate at $K{=}d_m{=}128$. (b) Four anchor sets of the same rank; B-0 and B-1 share condition number and orthogonality error yet differ in gain. Details in Appendices C.3.2 and C.3.1.

### 4.3 What Is Transferred

A transferred table could help for two reasons other than its content: the gain may be $\theta$'s own fitting of the domain, or the adapter rather than the anchor basis may carry the transfer. Two experiments test the two.

**Content attribution.** Fixing the injection rule and $\theta$ and changing only the frozen table (Table 2, Appendix C.1.5), a near-zero table still gains $+0.138$ on PubMed—about 40% of the full $+0.339$—so $\theta$ fits the domain without any memory. Random content adds $+0.06$, shuffled rows add $+0.02$, and restoring the slot-to-$n$-gram correspondence adds the largest step, $+0.12$. About 60% of the PubMed gain needs the trained table, and most of that needs the right rows under the right $n$-grams; on Math the trained table ($+0.071$) is below all three controls, the boundary case.

**Anchor rendering.** With $K < d_m$ anchors the rendering is rank-deficient (Figure 2a): although the adapter is trained as usual, the gain falls from $+0.120$ at $K = d_m$ to $+0.025$ at $K = 16$ while the direction error rises from $0.048$ to $0.52$, so the adapter does not recover what a deficient basis discards. Four anchor sets of the same rank (Figure 2b) give gains between $+0.026$ and $+0.089$; B-0 and B-1 share rank, condition number and orthogonality error and differ only in which tokens define the basis, yet gain $+0.026$ and $+0.067$. A random orthonormal basis of the same rank would decide whether an adapter alone suffices; that test is not part of the present experiments.

### 4.4 Ablation of the Injection Rule

Figure 3 removes one component at a time on the same frozen table. Two components decide whether a transferred table helps or hurts: without the gate the model is worse than with no memory in two of three settings, and without the adapter it is worse in all three. Differencing and RMS balancing change the gain by a few hundredths at the small table and less than one hundredth at the large one, and adding the rendered vector with a single learned scalar gives no measurable gain. Pruning at $f^* = n_{\text{ctx}}$ (C1) costs a median $0.06\%$ and at most $1\%$ of held-out loss against an oracle threshold over $48$ configurations, and the best threshold is stable across a tenfold change of learning rate (Appendix C.2).

### 4.5 Transfer to a Larger Target

Mounting the same table on the 14B backbone (Appendix C.4.2) gives a different picture: the 14B model starts from a lower loss in both domains and the gain contracts, from $+0.247$ to $-0.003$

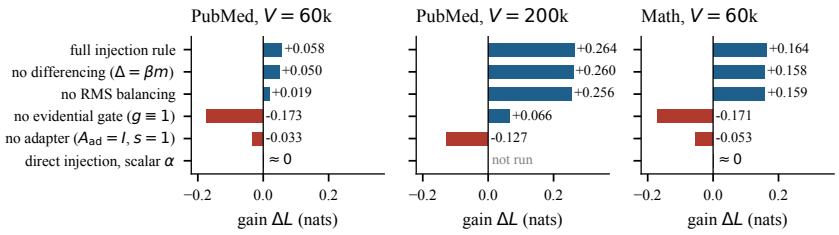


Figure 3: Gain $\Delta L$ with one component removed. All variants in a panel share one frozen 1.5B table (PubMed $V = 60$k: seed 42; PubMed $V = 200$k and Math: two and three seeds). Full numbers in Appendix C.1.

on PubMed and from $+0.150$ to $-0.149$ on Math. The contraction is not a single-seed artefact (Appendix C.4.2: 7B $+0.238 \pm 0.006$; 14B $+0.071 \pm 0.043$). Two readings are possible: the residual $\beta m - \hat{h}$ shrinks as the target already encodes what the table stores, or the 1.5B and 14B anchor subspaces are too far apart for the rendered directions to be usable. Content controls on 14B (Table 17) narrow them: emptying the table while keeping the injection rule and the training of $\theta$ gains $+0.127$ on PubMed and $+0.045$ on Math, random and shuffled tables give $-0.009$ and $+0.010$ on PubMed and $-0.255$ and $-0.167$ on Math, and the trained table does not beat the empty one. The controls do not settle which reading is right—1.5B→14B changes the scale and crosses a pre-training lineage at the same time—and three of the four control rows rest on a single seed.

## 5 Discussion and Limitations

**Where the gain stops.** HEMI's gain is largest where the target's own loss on the domain is highest (7B on all four domains) and does not survive on the 14B model, where the content controls of Section 4.5 place the failure on the target side rather than in the table; within one lineage (1.5B→7B) the gain stays positive in all four domains.

**Limitations.** Transfer is not free of training: $\theta$ is trained on the target with domain data, and on the source backbone its own fitting accounts for about 40% of the PubMed gain (Table 2). Every gain we report is within one model family and one pre-training lineage with a shared tokenizer; the one transfer that crosses a lineage (1.5B→14B) is also the one that fails (Section 4.5), and cross-family transfer is future work. Theorem 1 decomposes the direction error into three sources, but we measure only their sum, and the anchor term $\delta_{\max}$ is not calibrated on real backbones. The critical-frequency rule is validated only where few slots fall below the threshold; a sparse-regime sweep testing the predicted optimum $V^\star$ has not been run. Four baselines are missing and not claimed: zero-shot mounting, a table trained directly on the target ($V \times d_m + \theta \approx 7.75$M), full fine-tuning, and LoRA-based transfer (Wang et al., 2024; Farhadzadeh et al., 2025). The cost advantage is stated in trainable parameters and inference overhead, not in measured compute. Evaluation is by validation loss only, and out-of-domain retention matters here: the hash has no fingerprint, and only the gate stops a colliding out-of-domain $n$-gram from being injected. The Engram baselines of Appendix C.5 all use one reduced budget (1k samples, 2 epochs); a larger budget might reverse their signs.

## 6 Conclusion

HEMI moves domain knowledge out of a backbone's weights and into a hashed $n$-gram table that any backbone of the family and pre-training lineage reads through its own anchor basis. On DeepSeek-R1-Distill-Qwen, a table trained on the 1.5B model improves the 7B model in four domains with 65,811 parameters trained on the target and no extra forward pass at inference. Two questions remain open: whether the contraction on the 14B model reflects the target's smaller domain gap or a rendering mismatch across pre-training lineages, and whether a table transfers across model families.

### AI USE STATEMENT

In this work, we used generative AI tools for drafting and revising the text, for literature search, and for writing code documentation and figure scripts. We have not used generative AI tools to generate research ideas, to design experiments, to produce or alter experimental results, or to derive proofs; every number in the paper was obtained by running our own code. We have reviewed all AI-assisted work: AI-drafted passages were checked against the experiments and proofs by the authors, AI-suggested references were verified against the original sources, and AI-assisted code was tested by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

All experiments use publicly available backbones (DeepSeek-R1-Distill-Qwen 1.5B/7B/14B). Section 4.1 gives the configuration, the development/test split and the accounting conventions; Appendix E.1 gives the corpora and their preprocessing, Appendix E the implementation and hyper-parameter ranges of every baseline and Appendix E.3 the training hyper-parameters. The proof of Theorem 1 is in Appendix A. Code, data preparation scripts and configuration files will be released upon publication.

### ETHICS STATEMENT

This work studies parameter-efficient knowledge injection for language models. It involves no human subjects, personal data or sensitive attributes, and all corpora are public. We foresee no ethical risk beyond those generally associated with domain adaptation of language models.

## REFERENCES

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning (ICML)*, 2022.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2508.09874.

Xin Cheng, Rui Tian, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Chengqi Deng, Shangyan Zhou, Chenggang Zhao, Zhewen Hao, Yukun Li, Han Zhang, Zhengyan Zhang, Yixu Wei, M. Y. Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026.

Farzad Farhadzadeh, Debasmit Das, Shubhankar Borse, and Fatih Porikli. LoRA-X: Bridging foundation models with training-free cross-model adaptation. In *International Conference on Learning Representations (ICLR)*, 2025.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning LLMs on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*, 2024.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360. Association for Computational Linguistics, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Hugging Face. CodeParrot-clean: deduplicated Python files from GitHub. `https://huggingface.co/datasets/codeparrot/codeparrot-clean`, 2022.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13): 3521–3526, 2017.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Haodong Lei, Junming Liu, Yirong Chen, Pinlong Cai, Botian Shi, Ding Wang, and Hongsong Wang. TransMem: Transforming hidden states into memory for large language models. *arXiv preprint arXiv:2607.29032*, 2026.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

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

Mingyuan Li, Guangsheng Yu, Xu Wang, and Shaoxiong Ji. Cross-model memory transfer via target-side reader adaptation. *arXiv preprint arXiv:2608.17050*, 2026.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Jia Peng Lim and Hai Leong Chieu. Tokenizer-agnostic engram module. *arXiv preprint arXiv:2607.29065*, 2026.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2401.08565.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

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

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D. Manning. An emulator for fine-tuning large language models using small language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. In *International Conference on Learning Representations (ICLR)*, 2023.

National Library of Medicine. PubMed baseline repository, 2026 baseline. `https://ftp.ncbi.nlm.nih.gov/pubmed/baseline/`, 2026.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. CombLM: Adapting black-box language models through small fine-tuned models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.06786.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1333, 2023.

Ke Wang, Yiming Qin, Nikolaos Dimitriadis, Alessandro Favero, and Pascal Frossard. MEMOIR: Lifelong model editing with minimal overwrite and informed retention for LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Runqian Wang, Soumya Ghosh, David D. Cox, Diego Antognini, Aude Oliva, Rogério Feris, and Leonid Karlinsky. Trans-LoRA: Towards data-free transferable parameter efficient finetuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 61217–61237, 2024. arXiv:2405.17258.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation finetuning for language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2404.03592.

## GUIDE TO THE APPENDICES

The appendices are organised by the three components of HEMI, labelled as in Theorem 1 and Figure 1: C1, critical-frequency pruning (the finite-frequency term); C2, differential injection (the Stage-2 drift term); C3, anchor rendering (the anchor-mismatch term). Appendix A proves Lemmas 1 and 2 and Theorem 1, through the isometry of the rendering (C3), the signal–noise decomposition of a slot (C1) and the drift of the alignment pack (C2). Appendix B derives the critical frequency (C1), constructs the anchor set (C3), states the transfer protocol and its reference settings, and lists the gate features (C2). Appendix C reports the additional experiments: the injection rule and the attribution of the gain (C2), critical-frequency pruning (C1), anchor rendering (C3), the n-gram order and the larger target, and the numerical check of the theorem. Appendix D states the boundaries with the closest prior work and relates differential injection to knowledge editing; Appendix E gives baseline implementations.

# A FULL PROOF OF THEOREM 1

This appendix proves Lemmas 1 and 2 and Theorem 1 of Section 3.5. Three auxiliary results are used: the isometry of the rendering matrix (Lemma 3), the cross-model rendering error $\delta_{\max}$ (Theorem 2) and the signal–noise decomposition of a slot (Lemma 4). The three error sources are labelled as in the main text and in Figure 1: finite-frequency noise (C1), Stage-2 drift (C2) and anchor mismatch (C3).

**Notation and setup.** Let $M \in \mathbb{R}^{V\times d_m}$ be the frozen Stage-1 memory table. For slot $k$, $u_k$ is the coherent signal accumulated in Stage 1 and $v_k$ the content actually stored, with $\hat{u}_k = u_k/\|u_k\|$ and $\hat{v}_k = v_k/\|v_k\|$. $\Phi_S, \Phi_B \in \mathbb{R}^{D\times d_m}$ are the rendering matrices of the source and the target in the common ambient space of condition (i), whose construction is given after Theorem 2, and $\Pi = \Phi_B\Phi_B^\top$ is the orthogonal projection onto $\mathrm{range}(\Phi_B)$. $T_{\mathrm{ad}} = A_{\mathrm{ad}}\,\mathrm{diag}(s)$ is the alignment-pack map at the end of Stage 2. The intended direction is $p_S = \Phi_S\hat{u}_k$ and the injected direction $p_B = \Phi_B T_{\mathrm{ad}}\hat{v}_k$. For a nonzero vector $x$ we write $\bar{x} = x/\|x\|$. The cosine is invariant to positive scaling, so

$$1 - \cos(p_S, \Pi p_B) = 1 - \cos\left(p_S, \overline{\Pi p_B}\right) = \tfrac{1}{2}\left\|p_S - \overline{\Pi p_B}\right\|^2,$$

and the theorem is a bound on the chord between two unit vectors. Two elementary facts are used: the chord between unit vectors obeys the triangle inequality along any path of unit vectors, and normalization is 2-Lipschitz away from the origin,

$$\|\bar{x} - \bar{y}\| \le \frac{2\,\|x - y\|}{\max(\|x\|, \|y\|)}. \tag{10}$$

**Lemma 3 (isometry of the rendering matrix)** *If* $\mathrm{rank}(\Phi_B) = d_m$*, then* $\Phi_B^\top\Phi_B = I_{d_m}$*, and for any* $u, v \in \mathbb{R}^{d_m}$*:*

$$\langle \Phi_B u, \Phi_B v\rangle = u^\top v, \qquad \|\Phi_B u\| = \|u\|$$

*Consequently* $\Pi\Phi_B = \Phi_B$ *(the projection is the identity on* $\mathrm{range}(\Phi_B)$*), and* $\Pi p_B = p_B$*.*

**Proof.** $\Phi_B$ is the $Q$ factor of the thin QR decomposition of $E_B[\mathcal{S}]^\top \in \mathbb{R}^{d_h^{(B)}\times d_m}$. When the $d_m$ anchor columns are linearly independent, $Q^\top Q = I_{d_m}$, i.e., $\mathrm{rank}(\Phi_B) = d_m$. Hence $\Phi_B^\top\Phi_B = I_{d_m}$, giving $\langle \Phi_B u, \Phi_B v\rangle = u^\top\Phi_B^\top\Phi_B v = u^\top v$ and $\|\Phi_B u\|^2 = u^\top u$. Since $p_B = \Phi_B\hat{v}_k \in \mathrm{range}(\Phi_B)$, we have $\Pi p_B = \Phi_B\Phi_B^\top\Phi_B\hat{v}_k = \Phi_B\hat{v}_k = p_B$. □

**Theorem 2 (cross-model rendering error bound)** *Let* $\Phi_S$*,* $\Phi_B \in \mathbb{R}^{D\times d_m}$ *be the rendering matrices of the source and target models, taken in a common ambient space of dimension* $D$ *and both with orthonormal columns (*$\Phi_S^\top\Phi_S = \Phi_B^\top\Phi_B = I_{d_m}$*). Define the cross-model matrix* $W = \Phi_S^\top\Phi_B \in \mathbb{R}^{d_m\times d_m}$*. The singular values of* $W$ *are* $\sigma_i(W) = \cos\theta_i$*, where* $\{\theta_i\}$ *are the principal angles between* $\mathrm{range}(\Phi_S)$ *and* $\mathrm{range}(\Phi_B)$*. For any memory vector* $\hat{v}$*:*

$$\|\Phi_S\hat{v} - \Phi_B\hat{v}\| \le 2\|\hat{v}\|\sin(\theta_{\max}/2)$$

*where* $\theta_{\max} = \max_i\theta_i$ *is the largest principal angle. The maximum rendering mismatch over unit-norm directions is:*

$$\delta_{\max} = \max_{\|\hat{v}\|=1}\|\Phi_S\hat{v} - \Phi_B\hat{v}\| = 2\sin(\theta_{\max}/2)$$

**Proof.** For a unit vector $\hat{v}$:

$$\|\Phi_S\hat{v} - \Phi_B\hat{v}\|^2 = \|\Phi_S\hat{v}\|^2 + \|\Phi_B\hat{v}\|^2 - 2\hat{v}^\top\Phi_S^\top\Phi_B\hat{v} = 2 - 2\hat{v}^\top W\hat{v}$$

The symmetric part $(W + W^\top)/2$ has eigenvalues $\cos\theta_i$, so $\min_{\|\hat{v}\|=1}\hat{v}^\top W\hat{v} = \cos\theta_{\max}$, and therefore

$$\max_{\|\hat{v}\|=1} \|\Phi_S \hat{v} - \Phi_B \hat{v}\|^2 = 2(1 - \cos\theta_{\max}) = 4\sin^2(\theta_{\max}/2)$$

Taking square roots gives $\delta_{\max} = 2\sin(\theta_{\max}/2)$. The inequality for general $\hat{v}$ follows by homogeneity. □

**Remark (within-model isometry is not cross-model alignment).** Lemma 3 guarantees geometry preservation within a single model. The isometric embedding of a $d_m$-dimensional subspace into $\mathbb{R}^D$ is, however, unique only up to an orthogonal rotation: two models $S$ and $B$ can each render isometrically while their rendering subspaces are offset by the principal angles $\{\theta_i\}$. These angles are not a constant of the method; they depend on how consistent the embeddings of the anchor set are across the two models, which is why the anchor selection strategy (Appendix B.3) enters the bound through $\delta_{\max}$. The theorem is stated for a common ambient dimension $D$; the construction below is the common ambient space of condition (i) of Theorem 1. For backbones with different hidden sizes the two hidden spaces are identified through the anchor-token correspondence: the anchor rows of $S$ are aligned to those of $B$ by the orthogonal Procrustes map $U = \arg\min_{U^\top U=I} \|E_S[\mathcal{S}]\, U^\top - E_B[\mathcal{S}]\|_F$, $U \in \mathbb{R}^{d_h^{(B)} \times d_h^{(S)}}$, $\Phi_S$ is embedded as $U\Phi_S$, which keeps its columns orthonormal, and $\theta_{\max}$ is the largest principal angle between $\mathrm{range}(U\Phi_S)$ and $\mathrm{range}(\Phi_B)$. A block-diagonal embedding, in which the two hidden spaces are orthogonal to each other, would give $\theta_{\max} = 90^\circ$ and is excluded. Whenever $\theta_{\max}$ and $\delta_{\max}$ are evaluated they are computed in this space, and so is the direction error of Appendix C.3.2, with the stored direction as reference; no measured values of $\theta_{\max}$ on real backbones are reported in this paper (Section 3.2).

**Lemma 4 (signal–noise decomposition of a slot; $O(1/\sqrt{f_k})$ residual)** *Let $v_k^{(f_k)}$ be the memory vector after $f_k$ hits, decomposed as $v_k^{(f_k)} = s_k + \varepsilon_k$, where $s_k = \eta_V f_k u_k$ is the coherent signal component and $\varepsilon_k$ is the noise residual. Under Stage-1 hash-slot SGD updates (constant learning rate $\eta_V$) with isotropic initialization $v_k^{(0)} \sim \mathcal{N}(0, \sigma_V^2 I_{d_m})$, assume:*

- *(i) **Signal dilution**: each hit involves $n_{\mathrm{ctx}}$ context positions sharing the same slot direction $u_k$. Attention distributes weight across positions, so the total signal per hit is $\|u_k\|$ (independent of $n_{\mathrm{ctx}}$), not $n_{\mathrm{ctx}}\|u_k\|$.*
- *(ii) **Noise accumulation**: per-position gradient noise is independent across positions. The noise of a single hit, $N^{(t)} = \sum_{j=1}^{n_{\mathrm{ctx}}} n_j^{(t)}$, satisfies $\mathbb{E}\|N^{(t)}\|^2 = n_{\mathrm{ctx}} d_m \sigma_n^2$, where $\sigma_n^2$ is the per-position, per-dimension gradient noise variance.*
- *(iii) **Per-dimension** SNR: gradient noise and signal have comparable per-dimension magnitude, i.e., $\sigma_n^2 \asymp \|u_k\|^2/d_m$ (per-dimension gradient SNR is $\Theta(1)$). We assume this for LLM training, where the gradient signal is spread over many dimensions; it is not verified directly in this paper.*

*Then*

$$\mathbb{E}[\|\varepsilon_k\|^2] \le d_m\sigma_V^2 + n_{\mathrm{ctx}} d_m \sigma_n^2 \eta_V^2 f_k, \qquad q_k \triangleq \frac{n_{\mathrm{ctx}} d_m \sigma_n^2}{\|u_k\|^2},$$

$$\frac{\mathbb{E}[\|\varepsilon_k\|^2]}{\|s_k\|^2} \le \frac{q_k}{f_k}\Big(1 + \frac{\sigma_V^2}{n_{\mathrm{ctx}}\sigma_n^2\eta_V^2 f_k}\Big),$$

*where $q_k$ is the critical frequency $f^*$ of Section 3.4 at $\mathrm{SNR}_0 = 1$ (A2); under assumption (iii), which is (A1) of Section 3.4, $q_k \asymp n_{\mathrm{ctx}}$, and the proportionality constant is set to 1, which fixes $f^* = n_{\mathrm{ctx}}$ in all experiments. In the regime where gradient noise dominates the initialization floor, $n_{\mathrm{ctx}}\sigma_n^2\eta_V^2 f_k \ge \sigma_V^2$, the bracket is at most 2.*

**Proof.** The SGD iteration is $v_k^{(f_k)} = v_k^{(0)} + \eta_V f_k u_k - \eta_V \sum_{t=1}^{f_k} N^{(t)}$, where $N^{(t)} = \sum_{j=1}^{n_{\mathrm{ctx}}} n_j^{(t)}$ is the noise of a single hit. The signal is $s_k = \eta_V f_k u_k$ (deterministic, growing linearly in $f_k$). The residual is $\varepsilon_k = v_k^{(0)} - \eta_V \sum_t N^{(t)}$. Two noise components: initialization noise $\mathbb{E}\|v_k^{(0)}\|^2 = d_m\sigma_V^2$ (independent of $f_k$), and accumulated gradient noise $\mathbb{E}\|\eta_V \sum_t N^{(t)}\|^2 = n_{\mathrm{ctx}} d_m \sigma_n^2 \eta_V^2 f_k$ (a random walk over $f_k$ steps). Hence $\mathbb{E}\|\varepsilon_k\|^2 \le d_m\sigma_V^2 + n_{\mathrm{ctx}} d_m \sigma_n^2 \eta_V^2 f_k$.

Dividing by $\|s_k\|^2 = \eta_V^2 f_k^2 \|u_k\|^2$ gives the second inequality. The signal-to-noise ratio is $\mathrm{SNR}(f_k) = \|s_k\|^2/\mathbb{E}\|\varepsilon_k\|^2$; in the gradient-noise-dominant region it is $\approx f_k\|u_k\|^2/(n_{\mathrm{ctx}} d_m \sigma_n^2) = f_k/q_k$, so $\mathrm{SNR}(f^*) = 1$ gives $f^* = q_k$, and under assumption (iii) the factors $d_m$, $\sigma_n^2$ and $\|u_k\|$ cancel, $f^* \asymp n_{\mathrm{ctx}}$. □

**Proof of Lemma 2.** Fix $\rho \in (0,1)$. In the gradient-noise-dominant regime Lemma 4 gives $\mathbb{E}[\|\varepsilon_k\|^2]/\|s_k\|^2 \le 2q_k/f_k$, so by Markov's inequality, with probability at least $1-\rho$,

$$\frac{\|\varepsilon_k\|}{\|s_k\|} \le \sqrt{\frac{2\,q_k}{\rho\, f_k}}.$$

On this event apply (10) with $x = v_k = s_k + \varepsilon_k$ and $y = s_k$, whose normalizations are $\hat v_k$ and $\hat u_k$ ($s_k$ is a positive multiple of $u_k$):

$$\|\hat v_k - \hat u_k\| \le \frac{2\|\varepsilon_k\|}{\max(\|v_k\|, \|s_k\|)} \le \frac{2\|\varepsilon_k\|}{\|s_k\|} \le 2\sqrt{\frac{2}{\rho}}\sqrt{\frac{q_k}{f_k}} = \frac{\nu_f}{\sqrt{f_k}},$$
$$\nu_f = c_\rho \frac{\sqrt{n_{\mathrm{ctx}} d_m}\,\sigma_n}{\|u_k\|}, \qquad c_\rho = 2\sqrt{2/\rho},$$

since $\sqrt{q_k} = \sqrt{n_{\mathrm{ctx}} d_m}\,\sigma_n/\|u_k\|$. Under (A1), $q_k = n_{\mathrm{ctx}}$ and $\nu_f = c_\rho\sqrt{n_{\mathrm{ctx}}}$. Under Gaussian noise the concentration of $\|\varepsilon_k\|^2$ around its mean improves $c_\rho$ to $2\sqrt{2}\,(1+o(1))$ for large $d_m$; Markov's inequality suffices for the statement. With condition (iii) of Theorem 1, $f_k \ge c_0 f^* = c_0 q_k$ under (A2), the bound reads $\|\hat v_k - \hat u_k\| \le c_\rho/\sqrt{c_0}$. □

**Proof of Lemma 1.** Write the composite map as one matrix $T_{\mathrm{ad}} = A_{\mathrm{ad}}\,\mathrm{diag}(s) \in \mathbb{R}^{d_m \times d_m}$, initialized at $I$, and let $T^{(t)}$ be its value after $t$ SGD steps, $T^{(t+1)} = T^{(t)} - \eta\,\partial\mathcal{L}_t/\partial T_{\mathrm{ad}}$. By assumption (b) and then (a), $\|\partial\mathcal{L}_t/\partial T_{\mathrm{ad}}\|_F \le L_g\|g_t\Delta_t\| \le L_g G_{\max}$ at every step, so after $T_2$ steps

$$\|T^{(T_2)} - I\|_F \le \eta \sum_{t=1}^{T_2} \big\|\partial\mathcal{L}_t/\partial T_{\mathrm{ad}}\big\|_F \le \eta\, T_2\, L_g\, G_{\max},$$

and $\|\cdot\|_{\mathrm{op}} \le \|\cdot\|_F$ gives the operator bound. For the direction of a unit vector $x$, (10) with $\|x\| = 1$ gives $\|\overline{T_{\mathrm{ad}}x} - x\| \le 2\|T_{\mathrm{ad}}x - x\| \le 2\|T_{\mathrm{ad}} - I\|_{\mathrm{op}} \le 2\eta T_2 L_g G_{\max} = \nu_{\mathrm{drift}} G_{\max}$; the factor 2 in $\nu_{\mathrm{drift}} \triangleq 2\eta T_2 L_g$ is the Lipschitz constant of normalization. When $A_{\mathrm{ad}}$ and $s$ are updated separately rather than through $T_{\mathrm{ad}}$, the chain rule multiplies the gradient by factors bounded by $\|A_{\mathrm{ad}}\|_{\mathrm{op}}$ and $\|s\|_\infty$, which stay $O(1)$ over Stage 2 and are absorbed into $L_g$. □

**Proof of Theorem 1.** Work on the event of Lemma 2, which has probability at least $1-\rho$. By condition (ii) and Lemma 3, $\Phi_B$ is an isometry on $\mathbb{R}^{d_m}$: $\Pi p_B = p_B$, $\|\Phi_B x\| = \|x\|$ and $\overline{\Phi_B x} = \Phi_B \bar x$ for every nonzero $x$; in particular $\overline{\Pi p_B} = \Phi_B\,\overline{T_{\mathrm{ad}}\hat v_k}$. Insert two unit vectors of $\mathrm{range}(\Phi_B)$ between $p_S$ and $\overline{\Pi p_B}$ and apply the triangle inequality:

$$\big\|p_S - \bar p_B\big\| \le \underbrace{\|\Phi_S\hat u_k - \Phi_B\hat u_k\|}_{\text{anchor mismatch (C3)}} + \underbrace{\|\Phi_B\hat u_k - \Phi_B\hat v_k\|}_{\text{finite frequency (C1)}} + \underbrace{\big\|\Phi_B\hat v_k - \Phi_B\,\overline{T_{\mathrm{ad}}\hat v_k}\big\|}_{\text{Stage-2 drift (C2)}}.$$

*(C3)*: Theorem 2 applied to the unit vector $\hat u_k$ gives at most $\delta_{\max} = 2\sin(\theta_{\max}/2)$, where by condition (i) $\theta_{\max}$ is the largest principal angle between the two anchor subspaces in the common ambient space. *(C1)*: by the isometry the second term equals $\|\hat u_k - \hat v_k\|$, which is at most $\nu_f/\sqrt{f_k}$ by Lemma 2, whose regime is guaranteed by condition (iii). *(C2)*: by the isometry the third term equals $\|\hat v_k - \overline{T_{\mathrm{ad}}\hat v_k}\|$, which is at most $\nu_{\mathrm{drift}} G_{\max}$ by Lemma 1 under condition (iv). Summing the three, $\|p_S - \overline{\Pi p_B}\| \le \varepsilon_k$, and $1 - \cos(p_S, \Pi p_B) = \frac{1}{2}\|p_S - \overline{\Pi p_B}\|^2 \le \frac{1}{2}\varepsilon_k^2$. □

**Expansion of the bound.** Writing $a = \delta_{\max}$, $b = \nu_{\mathrm{drift}} G_{\max}$ and $c = \nu_f/\sqrt{f_k}$,

$$\tfrac{1}{2}\varepsilon_k^2 = \frac{\delta_{\max}^2}{2} + \frac{\nu_{\mathrm{drift}}^2 G_{\max}^2}{2} + \frac{\nu_f^2}{2f_k} + \nu_{\mathrm{drift}}\delta_{\max}G_{\max} + \frac{\nu_f\delta_{\max}}{\sqrt{f_k}} + \frac{\nu_f\nu_{\mathrm{drift}}G_{\max}}{\sqrt{f_k}}\ :$$

three squared terms, one per source, and three pairwise cross terms, of which $\nu_{\mathrm{drift}}\delta_{\max}G_{\max}$ does not decay with $f_k$. By $(a+b+c)^2 \le 3(a^2+b^2+c^2)$ the bound is at most $\frac{3}{2}(\delta_{\max}^2 + \nu_{\mathrm{drift}}^2 G_{\max}^2 + \nu_f^2/f_k)$, the form quoted in Section 3.5.

**Deterministic instantiation.** If, in place of the high-probability statement of Lemma 2, the residual ratio is assumed directly, $\|\varepsilon_k\|/\|s_k\| \le 1/\sqrt{c_0}$ with $c_0 = f_k/f^*$ (the construction of Appendix C.6), the (C1) chord is at most $2/\sqrt{c_0} = 2\sqrt{n_{\mathrm{ctx}}/f_k}$ under (A1), that is $\nu_f = 2\sqrt{n_{\mathrm{ctx}}}$; instantiating the drift chord as $\nu_{\mathrm{drift}} G_{\max} = 2\sqrt{2}\, G_{\max}$, the expansion becomes

$$\frac{2n_{\mathrm{ctx}}}{f_k} + \frac{4\sqrt{2}\, G_{\max}\sqrt{n_{\mathrm{ctx}}}}{\sqrt{f_k}} + 4G_{\max}^2 + \frac{\delta_{\max}^2}{2} + \frac{2\,\delta_{\max}\sqrt{n_{\mathrm{ctx}}}}{\sqrt{f_k}} + 2\sqrt{2}\,\delta_{\max} G_{\max},$$

which is the right-hand side checked numerically in Appendix C.6.

**Remark (reference direction).** $p_S$ is the rendering of the signal direction $\hat{u}_k$, not of the stored direction $\hat{v}_k$. Comparing the two renderings of the same stored vector, $\|\Phi_S \hat{v}_k - \Phi_B \hat{v}_k\| \le \delta_{\max}$ by Theorem 2, would not see finite-frequency noise at all: an under-trained slot is rendered consistently on both backbones, but consistently wrong. Measuring against $\hat{u}_k$ charges the slot for the noise it stores, which is why (C1) appears in the bound and why condition (iii) is needed.

### A.1 Asymptotic Analysis and Remarks on Theorem 1

The drift term $\frac{1}{2}\nu_{\mathrm{drift}}^2 G_{\max}^2$ is a floor independent of $f_k$: each gated update perturbs the alignment-pack map by a bounded amount, and Lemma 1 accumulates the perturbations linearly in $T_2$. It is controlled through $G_{\max}$: with $\nu_{\mathrm{drift}} = 2\sqrt{2}$ as instantiated in Appendix C.6, it is at most $0.09$ when $G_{\max} \le 0.15$. If $G_{\max}$ is scheduled to decay as $O(f_k^{-1/2})$, the floor and the coupling term $\nu_f \nu_{\mathrm{drift}} G_{\max}/\sqrt{f_k}$ are both $O(1/f_k)$, and for $\delta_{\max} \to 0$ the whole bound is $O(1/f_k)$; for a fixed $\delta_{\max}$ the terms $\frac{1}{2}\delta_{\max}^2$ and $\nu_{\mathrm{drift}}\delta_{\max} G_{\max}$ do not decay in $f_k$. For large $f_k$ the term $\nu_f^2/(2f_k)$ dominates the decaying part and the floor appears as a small residual plateau.

**Remarks.** The bound is safe, not tight: the terms are combined by the triangle inequality, and in the numerical verification of Appendix C.6 the ratio of empirical error to bound is at most 0.12 over the full grid, so the theorem characterizes how each perturbation source scales rather than predicting the error magnitude. Violating a condition changes the scaling rather than causing divergence: with insufficient rank the isometry of Lemma 3 fails and the cosine has no lower bound, with insufficient frequency the finite-frequency term $\nu_f/\sqrt{f_k}$ does not decay, and with unbounded injection the drift floor $\frac{1}{2}\nu_{\mathrm{drift}}^2 G_{\max}^2$ cannot be ignored; each case corresponds to one identifiable power law in Appendix C.6.

**Role of the conditions.**

- Insufficient rank (condition (ii), C3) $\to$ the image space of $\Pi$ has dimension $< d_m$, $p_B$ is flattened into a lower-dimensional subspace and the cosine has no lower bound (anchor-count sweep, Appendix C.3.2)
- Unbounded injection (condition (iv), C2) $\to$ the drift floor $\frac{1}{2}\nu_{\mathrm{drift}}^2 G_{\max}^2$ dominates and the guarantee fails (the no-gate variant of Appendix C.1)
- Insufficient frequency (condition (iii), C1) $\to$ initialization noise dominates $v_k$, the stored direction $\hat{v}_k$ and hence $p_B$ are random, and the cosine does not converge (pruning below $f^*$ does not degrade held-out loss, Appendix C.2)

Each condition controls one term of the bound; violating it removes the corresponding decay.

## B Method Details and Full Derivations

### B.1 Notation

Table 3 collects the notation of the main text. Appendix B.2 derives the critical frequency (C1); Appendix B.3 constructs the anchor set (C3); Appendix B.4 states the transfer protocol, its reference settings and its cost accounting; Appendix B.5 lists the gate features; the injection rule (C2) is analysed in Lemma 1 and ablated in Appendix C.1.

Table 3: Notation.

| | |
|---|---|
| $V, d_m, n$ | table size, memory dimension, n-gram order |
| $M \in \mathbb{R}^{V\times d_m}, v_k$ | memory table and the direction stored in slot $k$ |
| $E_B, \mathcal{S}, K = \lvert\mathcal{S}\rvert$ | embedding matrix of backbone $B$, anchor token set, its size |
| $\Phi_B \in \mathbb{R}^{d_h\times d_m}$ | rendering matrix of backbone $B$ (orthonormal columns) |
| $h, \hat{h}$ | hidden state at the injection layer and its projection into memory space |
| $\theta = \{\alpha, s, w, b, A_{\text{ad}}\}$ | alignment pack: magnitude, scale vector, gate weights and bias, adapter |
| $\beta, g$ | balance factor and gate value at one token |
| $u_k, \hat{u}_k, \hat{v}_k$ | coherent Stage-1 signal of slot $k$, its direction, and the direction of the stored content |
| $T_{\text{ad}} = A_{\text{ad}}\,\text{diag}(s)$ | alignment-pack map applied to a retrieved vector |
| $\nu_f, \nu_{\text{drift}}, \delta_{\max}$ | finite-frequency, drift and anchor-mismatch constants (Lemma 2, Lemma 1, Theorem 2) |
| $c_0, \theta_{\max}, G_{\max}$ | frequency margin $f_k/f^*$, largest principal angle between anchor subspaces, bound on the gated injection |
| $T, L, n_{\text{ctx}}$ | training samples, n-gram positions per sample, context positions sharing a slot at one step |
| $f_k, f^*, V^\star$ | hit count of slot $k$, critical frequency, table-size bound |

### B.2 Full Derivation of the Critical Frequency $f^*$

**Setting.** Consider a table of size $V$ trained on $T$ samples of $L$ n-gram positions each, so that $T \cdot L$ n-gram positions are hashed onto $V$ slots and the expected hit count of a slot is $\mathbb{E}[f_k] = T \cdot L/V$ (a Poisson approximation, valid for $V \ll |\text{vocab}|^n$). The hit count $f_k$ is set by the data and the table size and counts how often a slot is updated; the critical frequency $f^*$ derived below is the number of updates a slot needs to be reliably learned and is independent of $V$. Requiring a typical slot to satisfy $\mathbb{E}[f_k] \geq f^*$ gives the table-size bound $V \leq T \cdot L/f^*$ of Section 3.4.

**Step 1: saturating signal model.** The converged vector of slot $k$ is $v_k = s_k + \varepsilon_k$ with

$$s_k = \eta_V u_k \cdot (1 - \exp(-\lambda_{\text{sat}} f_k))/\lambda_{\text{sat}},$$

where $u_k$ is the fixed target direction of the slot, $\eta_V$ the table learning rate and $\lambda_{\text{sat}} > 0$ a saturation coefficient. For $\lambda_{\text{sat}} f_k \ll 1$ the signal reduces to the linear form $s_k = \eta_V f_k u_k$ of Lemma 4; for large $f_k$ it levels off at $\eta_V u_k/\lambda_{\text{sat}}$. The residual $\varepsilon_k = \varepsilon_V + \varepsilon_n$ has a static floor $\varepsilon_V$ from initialization, with $\mathbb{E}\|\varepsilon_V\|^2 = d_m\sigma_V^2$, and a gradient random-walk part $\varepsilon_n$ that grows with the hit count, $\mathbb{E}\|\varepsilon_n\|^2 = n_{\text{ctx}} d_m \sigma_n^2 \eta_V^2 f_k$; the two are independent, so $\mathbb{E}\|\varepsilon_k\|^2 = d_m\sigma_V^2 + n_{\text{ctx}} d_m \sigma_n^2 \eta_V^2 f_k$. This assumes a small $\eta_V$, weakly correlated gradient noise across steps, and independence of signal and noise, and is meant for low to moderate repetition counts.

**Step 2: signal-to-noise ratio.**

$$\text{SNR}(f_k) = \frac{\mathbb{E}\|s_k\|^2}{\mathbb{E}\|\varepsilon_k\|^2} = \frac{\eta_V^2 \|u_k\|^2 (1 - \exp(-\lambda_{\text{sat}} f_k))^2/\lambda_{\text{sat}}^2}{d_m\sigma_V^2 + n_{\text{ctx}} d_m \sigma_n^2 \eta_V^2 f_k}.$$

**Step 3: decision rule.** Whether a slot has been learned is a binary detection problem, $\mathcal{H}_0 : v_k = \varepsilon_k$ against $\mathcal{H}_1 : v_k = s_k + \varepsilon_k$. Under Gaussian noise the recall probability at false-positive rate FPR is that of a matched filter,

$$P_{\text{recall}} = Q\big(Q^{-1}(\text{FPR}) - \sqrt{\text{SNR}(f_k)}\big),$$

with $Q$ the Gaussian complementary distribution function. A target recall $P_0$ at a tolerated FPR therefore fixes a threshold $\text{SNR}_0$ through $P_0 = Q(Q^{-1}(\text{FPR}) - \sqrt{\text{SNR}_0})$, and the critical frequency is defined by $\text{SNR}(f^*) = \text{SNR}_0$; the common SNR $\geq 1$ rule is the choice of $(P_0, \text{FPR})$ for which $\text{SNR}_0 = 1$.

**Step 4: closed form.** Substituting the decision rule into Step 2 gives an implicit equation for $f^*$ with no closed-form solution in general. When gradient noise dominates the floor, $n_{\text{ctx}}\sigma_n^2\eta_V^2 f_k \gg \sigma_V^2$, and the signal is still in its linear regime, $\lambda_{\text{sat}} f^* \ll 1$, the floor term and the saturation drop out, $\text{SNR}_0 \approx f^*\|u_k\|^2/(n_{\text{ctx}} d_m \sigma_n^2)$, and

$$f^* = \text{SNR}_0 \cdot \frac{n_{\text{ctx}} d_m \sigma_n^2}{\|u_k\|^2},$$

which is the expression of Section 3.4. Outside this regime the implicit equation has to be solved numerically.

**Discussion of the assumptions.** The derivation rests on a saturating signal model and on a detection-theoretic threshold rather than on unbounded linear accumulation and a fixed SNR $\geq 1$; the saturating model reduces to the linear one at low repetition counts, and the SNR $\geq 1$ rule is the special

case of the detection rule in which $(P_0, \mathrm{FPR})$ give $\mathrm{SNR}_0 = 1$. The closed-form $f^*$ is a limiting approximation valid only within the stated regime; outside it the implicit equation must be solved numerically. The modeling assumptions are testable: the saturation of the memory signal with repetition count, the approximately linear growth of gradient noise with frequency, and the scaling of $f^*$ with context length and signal norm. Only the last is examined in this paper, and only indirectly (Appendix C.2).

### B.3 Construction of the Anchor Set $\mathcal{S}$

The quality of the rendering matrix $\Phi_B$ depends on the choice of anchor set $\mathcal{S}$, which must satisfy four criteria:

1. **Cross-model existence.** Every token in $\mathcal{S}$ must have a corresponding id in every backbone's vocabulary; otherwise $\Phi_B$ cannot be constructed on all backbones simultaneously.
2. **Stability across models.** We select tokens that are frequent in the mixed-domain corpus; in practice these are short function words, common sub-word pieces and frequent code identifiers (Strategy B-1 below), whose embeddings are expected to drift less across models than those of rare content words, which enlarge the mismatch between the two rendering subspaces.
3. **Linear independence.** The embedding rows of the $K$ anchors must be linearly independent to guarantee $\mathrm{rank}(\Phi_B) = d_m$.
4. **Sufficient cardinality.** $K \geq d_m = 128$, the rank condition of Theorem 1; for $K < d_m$ the rendered directions are confined to a subspace of dimension $K$ and the bound no longer applies (anchor-count sweep in Appendix C.3.2).

Four selection strategies are compared in Appendix C.3.1. *A (vocabulary prefix)* takes the first $K$ token ids, which include special symbols with weak semantics. *B* filters out special and non-ASCII tokens and ranks the rest, either by a length proxy (*B-0*: shorter tokens first, as a stand-in for frequency) or by *corpus frequency* (*B-1*: counts over a mixed PubMed/Code/General corpus of 100k lines per file, 17,401 stable tokens; sample anchors def, ING, a, param, th, in, The, math). *C (multi-model intersection)* ranks the string intersection of several vocabularies by merged frequency, which is strictly consistent across models but may be smaller than $K$. We adopt B-1.

### B.4 Transfer Protocol and Cost Accounting

**Three-stage protocol.** The protocol is stated in Section 3.2; Stage 1 saves the pruned table and $\theta$ once, and the pruned table is the transferable asset.

**Reference settings.** Two settings deviate from the Stage-2 protocol and serve as reference points. *Zero-shot*: the frozen table and $\Phi_B$ are mounted and $\theta$ is left at its initialization, so nothing is trained on the target; it isolates what rendering alone provides. *Train-from-scratch*: the table and $\theta$ are both trained on the target backbone, at a target-side cost of $V \times d_m + |\theta|$ parameters; it is the reference against which the Stage-2 cost $|\theta|$ is counted. Neither setting is used for the HEMI rows of Section 4, and transfer without training $\theta$ on the target is not evaluated in the present experiments (Appendix C.4.2).

**Cost accounting.** Stage 2 trains $|\theta|$ parameters, against $V \times d_m + |\theta|$ for a table trained on the target (the train-from-scratch setting). These are counts of trainable parameters, not end-to-end compute: wall-clock cost also depends on forward passes and data loading, and the inference-time cost of each method is reported separately in Table 1.

### B.5 Gate Features

The evidential gate of Section 3.3 is $g = \sigma(w^\top e + b) \cdot \mathrm{mask}$ with $e \in \mathbb{R}^8$; $w$ and $b$ belong to $\theta$ and are trained in Stage 1 on the source and again in Stage 2 on the target. The eight features come from three sources.

- **Token-level features** (computed at every position in the forward pass). $e_1$: the cosine between $\beta m$ and $\hat{h}$, the two inputs of the residual $\Delta = \beta m - \hat{h}$. $e_2$: the running hit rate,

Table 4: Configuration of each experiment. Only settings that the experiment itself reports are listed; the protocol of each experiment is the one described where its table appears. "Validation" is the evaluation set; the main table uses the test part of Section 4.1. Base losses are PubMed / Math on the target backbone. A dash marks an entry that does not exist for that table, "n.r." a setting that was not recorded for that run.

| Table | Experiment | Backbone | $V$ | Train samples | Validation | Base (PubMed / Math) |
|---|---|---|---|---|---|---|
| 1, 15 | main comparison, cross-scale | 1.5B→7B, 14B | 60k | 12k (Code 5k) | test part, 450 | 2.868 / 2.335; 14B: 2.259 / 2.276 |
| 5 | PubMed ablation | 1.5B→7B | 60k | n.r. | n.r. | 3.478 / – |
| 6 | PubMed ablation, large table | 1.5B→7B | 200k | 20k | n.r. | 2.868 / – |
| 7 | Math ablation | 1.5B→7B | 60k | n.r. | n.r. | – / 2.559 |
| 2, 8 | attribution | 1.5B | 200k | 1k | n.r. | n.r. / 2.31 |
| 9 | pruning threshold | n.r. | n.r. | n.r. | n.r. | – |
| 10 | threshold vs. learning rate | n.r. | 200k | 5k | n.r. | – |
| 12 | anchor strategies | 1.5B→7B | 32k | n.r. | n.r. | 4.906 / – |
| 13 | anchor-count sweep | 1.5B→7B | 20k | 12k | n.r. | n.r. |
| 14 | n-gram order | 1.5B→7B | n.r. | n.r. | n.r. | n.r. |
| 16 | multi-seed transfer | 1.5B→7B, 14B | n.r. | n.r. | n.r. | 7B 2.826, 14B 2.224 (PubMed) |

the fraction of positions so far in the sequence whose trailing n-gram maps to a retained slot. $e_3$: the hash-dispersion bit count of the current n-gram [state the definition used in the code].

- **Stage-1 slot statistics** (recorded once per slot while the table is trained on the source and stored with the table). $e_4$: the log hit count $\log(1 + f_k)$. $e_5$: the variance of the gradient directions that colliding n-grams contributed to the slot, a measure of collision conflict. $e_6$: the pointwise mutual information between the n-gram and the token that follows it. $e_7$: [fourth slot statistic; supply from the code].
- **Per-token miss rate.** $e_8$: the fraction of positions so far whose slot was pruned at $f^*$, so that retrieval returned no content.

The cosine lets the gate close where memory and hidden state disagree; the slot statistics let it distrust slots that were hit rarely or averaged over conflicting n-grams; the hit and miss rates calibrate the gate to how much of the current sequence the table covers. The ablations of Appendix C.1 remove the gate as a whole (variant C, $g \equiv 1$); the features are not ablated individually.

# C Additional Experimental Results

Each experiment below is tagged with the component of HEMI that it tests, labelled as in Theorem 1 and Figure 1: C1, critical-frequency pruning; C2, differential injection; C3, anchor rendering.

Base losses differ between tables because the protocols differ; gains are comparable only within a table (Section 4.1) and only gains should be compared across tables. Table 4 lists the settings that each experiment itself reports; a setting that was not recorded for that run is marked "n.r.".

## C.1 Ablations of the Injection Rule

The ablations test C2, the injection rule: they remove one component of the injection rule at a time while keeping all other hyper-parameters fixed. Within each configuration all variants share **the same frozen Stage-1 table** $M$, so that only the injection rule changes. Appendices C.1.1–C.1.4 report validation loss on the 7B backbone and the gain $\Delta L = L_{\text{Base}} - L_{\text{variant}}$ relative to the no-memory baseline; Appendix C.1.5 is on the 1.5B backbone.

### C.1.1 PubMed Ablation (Table 5)

Two components decide the sign of the gain: removing the evidential gate changes it from $+0.058$ to $-0.173$ and removing the adapter to $-0.033$. The gate decides at which positions memory is injected; the adapter re-expresses memory vectors trained on the source backbone in the coordinates of the target. Without RMS balancing the gain is $+0.019$ and without differencing $+0.050$. The contributions cannot be added because the components interact; the ranking (gate, adapter, RMS

Table 5: Ablation of the injection rule on PubMed with a shared frozen Stage-1 table ($V = 60$k, 7B backbone).

| Variant | Modification | val loss | $\Delta L$ | % of Full |
|---|---|---|---|---|
| Base | No memory table | 3.478 | 0.000 | – |
| A. Full injection rule | Differencing + gate + RMS + adapter | 3.420 | $+0.058$ | 100% |
| B. × no differencing | Use $\Delta = m$ directly instead of $m - \hat{h}$ | 3.428 | $+0.050$ | 86% |
| C. × no evidential gate | Fix $g_i \equiv 1$, remove w from $\theta$ | 3.651 | $-0.173$ | $-298\%$ |
| D. × no RMS balancing | Normalize m to unit RMS (no hidden-state matching) | 3.459 | $+0.019$ | 33% |
| E. × no adapter | $A_{\text{ad}} = \text{I}$, $s = 1$ | 3.511 | $-0.033$ | $-57\%$ |

Table 6: Ablation of the injection rule under the high-data/large-table regime ($V = 200$k, 20k training samples), means over seeds $\{42, 456\}$.

| Variant (Stage 1: $V = 200$k, 20k samples) | val loss | $\Delta L$ | % of Full |
|---|---|---|---|
| Base | 2.8678 | 0.000 | – |
| A. Full injection rule | 2.6034 | $+0.2644 \pm 0.0003$ | 100% |
| B. × no differencing | 2.6083 | $+0.2595 \pm 0.0061$ | 98% |
| D. × no RMS | 2.6115 | $+0.2563 \pm 0.0128$ | 97% |
| C. × no evidential gate | 2.8018 | $+0.0659 \pm 0.0014$ | 25% |
| E. × no adapter | 2.9950 | $-0.1272 \pm 0.0386$ | $-48\%$ |

balancing, differencing) holds in this small-table, low-data configuration and its dependence on table size and data volume is examined next.

### C.1.2 PubMed at $V = 200$k

To test whether the ranking depends on table size and data volume, the ablation is repeated on PubMed with $V = 200$k and 20k training samples, a different Stage-1 table, and means over seeds 42 and 456:

Both seeds give the same ranking. Removing the adapter is the only variant with a negative gain, removing the gate reduces the gain to about a quarter, and removing differencing or RMS balancing changes it by less than one hundredth. The adapter is therefore the component without which injection hurts in both configurations; removing the gate degrades the gain in both and turns it negative only in the low-data one. With a larger table and more data the slots converge more cleanly and the memory vectors are closer in scale to the hidden state, so subtracting the projected hidden state and rescaling change less, and the remaining bottleneck is the alignment between source and target coordinates, the adapter's role.

### C.1.3 Direct Injection with a Single Scalar

To measure the injection rule as a whole rather than component by component, we implement the direct injector of Section 3.3, $h \leftarrow h + \alpha\beta\,\Phi_B v_k$, with none of its components: no differencing (the full memory vector is added), no evidential gate and no adapter, and the balance factor $\beta$ only puts the rendered vector on the scale of $h$, so that the only trained parameter is the scalar $\alpha$; the scale factor is kept in this baseline so that the comparison isolates the injection rule and not the magnitude. On both PubMed and Math this baseline obtains $\Delta L$ between $+0.0001$ and $+0.0002$ and under some seeds degrades the backbone: adding the frozen memory table with a single learned scalar provides no measurable benefit. The raw rendered memory vector therefore contributes no measurable gain by itself; the injection rule is what makes the transferred table usable.

Table 7: Ablation of the injection rule on Math, averaged over three random seeds.

| Variant | Modification | val loss | $\Delta L$ | % of Full |
|---|---|---|---|---|
| Base | No memory table | 2.559 | 0.000 | – |
| A. Full injection rule | Differencing + gate + RMS + adapter | 2.395 | $+0.164$ | 100% |
| B. × no differencing | Use $\Delta = m$ directly | 2.401 | $+0.158$ | 96% |
| C. × no evidential gate | Fix $g_i \equiv 1$ | 2.730 | $-0.171$ | $-104\%$ |
| D. × no RMS balancing | Unit RMS normalization only | 2.400 | $+0.159$ | 97% |
| E. × no adapter | $A_{\text{ad}} = \text{I}$, $s = 1$ | 2.612 | $-0.053$ | $-32\%$ |

Table 8: Attribution of the HEMI gain: memory-table content versus alignment-pack fitting. Mean ± std over seeds (42, 123, 456).

| Memory table content | PubMed $\Delta L$ | Math $\Delta L$ | Interpretation |
|---|---|---|---|
| [O] zero (content $\approx 0$) | $+0.1375 \pm 0.0050$ | $+0.0865 \pm 0.0042$ | Parameter-pack fitting only, no memory information |
| [R] random (no domain knowledge) | $+0.2011 \pm 0.0156$ | $+0.1023 \pm 0.0293$ | Random direction injection |
| [S] shuffled (real but permuted) | $+0.2210 \pm 0.0168$ | $+0.1284 \pm 0.0331$ | True statistics, slot correspondence lost |
| Full memory (HEMI) | $+0.3393 \pm 0.0052$ | $+0.0705 \pm 0.0038$ | Real trained table + correct slot correspondence |

### C.1.4 Cross-Domain Validation on Math (Table 7)

The same ablation on Math, averaged over seeds 42, 123 and 456:

Math gives the same ranking: removing the gate or the adapter yields a negative gain, removing RMS balancing or differencing changes the gain only slightly. The gain of the full rule ($+0.164$) is larger than on PubMed at the same table size because Math text repeats operators and symbolic n-grams that the table captures precisely.

### C.1.5 Attributing the Gain: Memory Content versus Alignment-Pack Fitting

Could the gain be the fitting ability of the alignment pack $\theta$ on the domain data, independent of the table content (C1)? This experiment keeps the injection rule and $\theta$ unchanged and replaces only the content of the frozen table $M$, with three settings: **[O] zero**: content set to $\mathcal{N}(0, 10^{-8})$ noise, so that the memory content has $\text{SNR} \approx 0$ (lower bound); **[R] random**: content randomly initialized (no domain knowledge); **[S] shuffled**: the Stage-1 table with its rows permuted, which keeps the statistics of the content but breaks the correspondence between slots and n-grams. The configuration is separate from the main table ($V = 200$k, 1k samples, $n = 3$, seeds 42, 123 and 456); the baseline and the four settings share one evaluation protocol, so the gains are comparable within the table. Table 8 reports mean ± std on PubMed and Math:

**Attribution.** (a) The alignment pack has a nonzero floor: with near-zero table content the PubMed gain is still $+0.138$, which does not depend on any memory knowledge. (b) Memory content adds to it: shuffled content improves over zero by $+0.084 \pm 0.022$ on PubMed, and the trained table improves over shuffled content by about $+0.12$ ($+0.339$ against $+0.221$), so the largest single contribution comes from the correct correspondence between slots and n-grams rather than from the statistics of the table alone. (c) On Math the four settings are close and the trained table gives the lowest gain ($+0.071$ against $+0.128$ shuffled and $+0.102$ random). The Math baseline in this configuration (2.31) is close to the domain floor, so there is little for the table to add; that the uninformative controls score higher indicates that $\theta$ and the gate can fit the validation distribution when the table carries no usable signal, which held-out loss alone cannot separate from genuine gain. (d) The

Table 9: Validation-loss degradation of pruning at $f^* = n_{\text{ctx}}$ relative to the oracle-optimal threshold, over 48 valid configurations in 3 domains.

| Domain | # configs | Median degradation | Max degradation | All <1%? |
|---|---|---|---|---|
| PubMed | 18 | 0.06% | 0.65% | ✓ |
| Math | 12 | 0.08% | 0.91% | ✓ |
| Code | 18 | 0.07% | 0.99% | ✓ |
| All | 48 | 0.06% | 0.99% | ✓ |

Table 10: Slots removed at each pruning threshold (out of $V = 200$k) and the best threshold under each Stage-1 learning rate ($\eta_V = 2 \times 10^{-4}$ / $2 \times 10^{-3}$), on three domains; $\bar{f}$ is the mean hit count per slot.

| Domain | $\bar{f}$ | Removed at threshold 100 | 150 | 200 | Best threshold (low / high $\eta_V$) |
|---|---|---|---|---|---|
| PubMed | 156.25 | 247 | 5,922 | 7,780 | 200 / 100 |
| Math | 156.25 | 211 | 5,497 | 7,609 | 200 / 200 |
| Code | 156.25 | 1,311 | 6,034 | 7,267 | 200 / 100 |

source-side decomposition on PubMed is therefore roughly $+0.14$ from the alignment pack and $+0.20$ from memory content. Table 2 in the main text reports the same numbers; the random-$\Phi_B$ control is discussed in Section 4.3.

## C.2 Critical-Frequency Pruning

### C.2.1 Pruning Threshold Validation (Table 9)

The pruning experiments test C1. Table 9 reports the validation loss degradation of pruning at $f^* = n_{\text{ctx}}$ relative to the oracle-optimal threshold, across 48 valid $(\sigma_V, \eta_V)$ configurations in 3 domains.

We sweep 20 $(\sigma_V, \eta_V)$ grid points per domain ($4\,\sigma_V \times 5\,\eta_V$). Of the 60 total points, 48 produce a valid oracle $f^*$ (the remaining 12 are degenerate configurations in which the trained table carries no information and no oracle threshold exists). For all 48 valid points, pruning at $f^* = n_{\text{ctx}}$ [state the value of $n_{\text{ctx}}$, hence of $f^*$, used here] yields a validation loss degradation of <1% relative to the oracle-optimal threshold, confirming that $f^* = n_{\text{ctx}}$ is a safe pruning threshold in this regime; the fraction of slots it removes is quantified in Appendix C.2.2.

### C.2.2 Stability of the Pruning Threshold across Learning Rates

The set of slots below a given hit count is fixed by the data and the hash function and does not depend on the learning rate. What can depend on it is the threshold at which pruning becomes harmless, since the learning rate changes how far a slot has converged at a given hit count. Tables with identical data, $V = 200$k and $n = 3$ are therefore trained at two learning rates ten times apart, $\eta_V \in \{2 \times 10^{-4}, 2 \times 10^{-3}\}$, on PubMed, Math and Code (5k samples), and Table 10 reports the slots removed at each threshold and the best threshold under each learning rate.

The pruned counts are identical under both learning rates, as they must be; the informative column is the last: the threshold with the best validation loss lies in $[100, 200]$ under both learning rates in all three domains (on Code the two optima differ by one step and the gains by 0.009), so the pruning rule does not need re-tuning when the Stage-1 learning rate changes within this range. In this regime few slots fall below the threshold: thresholds of 100 to 200 remove between 0.1% and 3.9% of the 200k slots, so the validation here and in Appendix C.2.1 covers the dense regime only; a sparse-regime sweep, in which most slots would be pruned, is left open (Section 5).

## C.3 Anchor Rendering

The two experiments below test C3.

Table 11: Anchor set quality for the four selection strategies.

| Strategy | Special-symbol ratio | Cross-model token consistency | Mean token length | Sample tokens |
|---|---|---|---|---|
| A (prefix) | 24.22% | 100% | 1.00 | !, ", #, $, %, & |
| B-0 (length proxy) | 0% | 100% | 1.97 | I, i, a, A, di, GE, oe |
| B-1 (corpus frequency) | 0% | 100% | 3.11 | def, ING, 0LD, a, param, th, in |
| C (intersection) | 0% | 100% | 1.97 | A, a, i, I, Tr, zk, Tx |

Table 12: End-to-end transfer performance of the four anchor strategies (PubMed, 1.5B → 7B).

| Strategy | Stage-1 val loss | Stage-2 val loss | Stage-2 gain | Base |
|---|---|---|---|---|
| A (prefix) | 5.3745 | 4.8799 | +0.0259 | 4.9058 |
| B-0 (length proxy) | 5.2004 | 4.8798 | +0.0260 | 4.9058 |
| B-1 (corpus frequency) | **5.3391** | **4.8392** | **+0.0666** | 4.9058 |
| C (intersection) | 5.3773 | 4.8173 | +0.0885 | 4.9058 |

### C.3.1 Comparison of Anchor Selection Strategies

We compare the four anchor selection strategies of Appendix B.3 (A vocabulary prefix, B-0 length proxy, B-1 corpus frequency, C multi-model intersection) with $K = 128 = d_m$, $V = 32$k, on PubMed, transferring a table trained on 1.5B to 7B. Table 11 describes the anchor sets and Table 12 the end-to-end transfer gain.

B-0 selects anchors that are short (mean 1.97) and mostly BPE fragments (GE / oe / eb) rather than semantically stable function words; B-1, by ranking on true corpus frequency, selects tokens that are longer (3.11) and semantically richer (def / param / The / math / left / right). Strategy A contains 24.22% special symbols and has weak semantic representativeness.

B-1 gains $+0.067$ against $+0.026$ for both A and B-0 and $+0.089$ for C. B-0 and B-1 share the filtering and sorting framework and differ only in the ordering criterion, token length against corpus frequency, and their rendering matrices have identical rank, condition number and orthogonality error, so the difference in gain comes from which tokens define the basis: B-1's anchors are longer (mean 3.11 against 1.97 characters) and semantically stable (def, param, The, math), whereas B-0's are mostly BPE fragments (GE, oe, eb). C gains most, but its anchors come from a vocabulary intersection that is smaller and more complex to build; B-1 is the compromise adopted. Two independent B-1 runs with different training seeds give $+0.0666$ and $+0.0650$.

### C.3.2 Anchor-Count Sweep

This sweep tests the rank condition (ii) of Theorem 1 (C3). We sweep the anchor set cardinality $K \in \{16, 32, 64, 128, 256\}$, rerunning Stages 0 to 2 for each $K$ while keeping $d_m$ fixed at 128. The rank of $\Phi_B$ equals $\min(K, d_m, \mathrm{rank}(E_B[\mathcal{S}]))$ and saturates at $d_m = 128$ once $K \geq 128$.

**What is measured.** The signal direction $\hat{u}_k$ of Theorem 1 is not observable, so the reported error takes the stored direction $\hat{v}_k$ as the reference on both backbones, $1 - \cos(\Phi_S \hat{v}_k,\ \Pi\, \Phi_B T_{\mathrm{ad}} \hat{v}_k)$ in the common space of Appendix A; it therefore contains the drift and mismatch terms (C2, C3) of the bound but not the finite-frequency term (C1). All three choices above—the reference direction, the application of $T_{\mathrm{ad}}$, and the space of the cosine—are made by the released evaluation script and are fixed across every reported number.

Two regimes are visible, consistent with Theorem 1:

1. **Rank-deficient regime** ($K < 128$): the error $1 - \cos$ is large ($0.52 \to 0.18$) and decreases only linearly in $K$; the gain scales sublinearly. Geometric reason: the image $\Pi p_B$ is flattened into a $K < d_m$ subspace, so even a perfect memory direction cannot be faithfully reproduced on the large backbone.

Table 13: Anchor-count sweep (1.5B→7B, PubMed, $V = 20$k, 12,000 samples): transfer direction error and HEMI gain versus anchor cardinality $K$.

| Number of anchors $K$ | 16 | 32 | 64 | 128 ($= d_m$) | 256 |
|---|---|---|---|---|---|
| Mean $\mathrm{rank}(\Phi_B)$ | 16 | 32 | 64 | 128 | 128 (saturated) |
| $1 - \cos(p_S, \Pi p_B)$ | 0.52 | 0.36 | 0.18 | 0.048 | 0.046 |
| HEMI $\Delta L$ | $+0.025$ | $+0.060$ | $+0.098$ | **+0.120** | $+0.122$ |
| Rank condition met? | $\times$ | $\times$ | $\times$ | ✓ | ✓ |

Table 14: HEMI gain versus n-gram order across four domains.

| Domain | n=2 | n=3 | n=4 | n=5 |
|---|---|---|---|---|
| PubMed | **+0.161** | $+0.125$ | $+0.121$ | $+0.123$ |
| Math | **+0.125** | $+0.114$ | $+0.101$ | $+0.104$ |
| Code | **+0.240** | $+0.173$ | $+0.146$ | $+0.153$ |
| General | **+0.199** | $+0.181$ | $-$ | $-$ |

2. **Rank-saturated regime** ($K \geq 128$): the error drops by nearly a factor of four, to about 0.048, between $K = 64$ and $K = 128$; the gain essentially saturates (only 2% difference between 128 and 256 anchors). $K = d_m = 128$ is therefore the smallest anchor count that satisfies the rank condition.

**Seeds.** In the single-seed sweep General gained less at $K = 128$ ($+0.037$) than at $K = 64$ ($+0.062$) and Math was slightly negative at $K = 64$ ($-0.005$); rerunning $K \in \{64, 128\}$ with seeds 123 and 456 gives $K = 128 > K = 64$ in all four runs (General $+0.102$ and $+0.112$ against $+0.068$ and $+0.061$; Math $+0.091$ and $+0.081$ against $+0.037$ and $+0.031$), so the seed-42 anomalies are within seed-to-seed variation.

## C.4 N-GRAM SWEEP AND CROSS-SCALE TRANSFER

### C.4.1 N-GRAM ORDER

The n-gram order sweep measures the sensitivity of the gain to the order $n$ on the 7B target. Across the four domains the gain decreases with $n$:

Bigrams give the highest gain in every domain, trigrams are close behind, and the gain falls for $n \geq 4$: longer n-grams occur less often, more slots fall below $f^*$, and fewer entries survive pruning. The order $n = 3$ was fixed before this sweep as a balance between raw gain and the longer context that trigrams capture; the advantage of $n = 2$ is a finding of the sweep, not a design choice, and suggests that phrase-level lexical patterns dominate these domains.

### C.4.2 CROSS-SCALE TRANSFER

This experiment asks where transfer stops (Section 4.5). We transfer the same Stage-1 memory table $M$ (trained on 1.5B) to the 7B and 14B backbones and measure the HEMI gain. The gain is read against the target model's own baseline loss:

The 14B baseline loss is lower than 7B's in both domains, and the gain contracts to $-0.003$ on PubMed and $-0.149$ on Math. Section 4.5 gives the two readings of this contraction, a smaller domain gap under the injection rule or a rendering mismatch across pre-training lineages, and reports the content controls on the 14B target (Table 17) that narrow them without settling which reading is right.

To add seed variance to the single-seed numbers above, we sweep seeds (42, 123, 456) under a separate 1.5B-table-to-target transfer protocol (baseline 7B 2.826, 14B 2.224). The protocol differs from that of Table 15, so only the seed variance, not the absolute gains, should be compared across the two tables:

Table 15: Cross-scale transfer of one Stage-1 table (trained on 1.5B) to the 7B and 14B targets.

| Domain | 7B baseline | 7B $\Delta L_{\mathrm{HEMI}}$ | 14B baseline | 14B $\Delta L_{\mathrm{HEMI}}$ |
|---|---|---|---|---|
| PubMed | 2.868 | $+0.247$ | 2.259 | $-0.003$ |
| Math | 2.335 | $+0.150$ | 2.276 | $-0.149$ |

Table 16: Multi-seed robustness of the cross-scale transfer gain under a dedicated 1.5B-to-target protocol.

| Target | seed 42 | seed 123 | seed 456 | mean $\pm$ std |
|---|---|---|---|---|
| 7B | – | $+0.242$ | $+0.234$ | $+0.238 \pm 0.006$ |
| 14B | $+0.022$ | $+0.098$ | $+0.092$ | $+0.071 \pm 0.043$ |

**Transfer variance.** On the 7B target the two available seeds agree to within 0.01 ($+0.234$ / $+0.242$) and match the main-table value ($+0.247$). On the 14B target the mean gain is about one third of 7B's ($+0.071$) and the seed spread is seven times larger (std 0.043; seed 42 gives $+0.022$, seeds 123 and 456 about $+0.09$). Larger relative variance at a smaller gain is consistent with the injected signal approaching the noise floor as the domain gap narrows, but with three seeds it does not by itself distinguish that explanation from seed noise.

Content controls on the 14B target test the two readings of Section 4.5. The protocol and the seed of Table 16 are kept and the same frozen 1.5B table is used throughout; only its content changes, as in Table 2 on the source backbone. Baseline losses are 2.224 (PubMed) and 2.261 (Math).

The ordering is the reverse of the one on the source backbone. On 1.5B the trained table is above all three controls (Table 2: $+0.339$ against $+0.138$, $+0.201$ and $+0.221$ on PubMed); on 14B it is below all three: the empty table gains $+0.127$ on PubMed and $+0.045$ on Math, and randomizing or shuffling the content costs a further $0.12$ to $0.30$ nats. The content of the table therefore contributes nothing positive on the 14B target, and the gain that remains is the alignment pack fitting the target's own distribution. What the controls do not settle is the cause: 1.5B$\rightarrow$14B changes the scale and crosses a pre-training lineage at the same time, and a rendering mismatch and a target that has absorbed the domain predict the same ordering here. Separating them would need a lineage-matched target of comparable strength, which we do not have; three of the four rows also rest on a single seed.

**Relation to the upgrade scenario.** The protocol above takes a table trained on one backbone and mounts it on another without retraining the table; only the alignment pack $\theta$ is trained on the target (Section 3.2). Transfer without training $\theta$ on the target is not evaluated here. The $1.5\mathrm{B} \rightarrow 7\mathrm{B}/14\mathrm{B}$ setting is a scale change within one model series on public domains, and Section 5 discusses the two readings of the smaller gain on the stronger target; it does not speak to cross-vendor upgrades on private distributions.

### C.5 Engram Baselines on the 7B Target

Table 18 implements the Engram injection rule—one vector retrieved per token by hashing the n-gram and added to the hidden state of a designated layer—and mounts it on the 7B target. Every row shares one reduced budget (1k training samples, 2 epochs, $V = 20$k, $d_m = 128$, injection at layer 10) and is compared against the others; the last row runs HEMI under the same budget and with the same frozen table. $\Delta L$ is measured against each backbone's own baseline (1.5B 3.134, 7B 2.826).

Read down the rows, the Engram variants form a monotone pattern in how much of the injection path is trained: injecting with nothing trained raises the loss by $11.5$ to $12.2$ nats (E3), training the table and projection natively leaves about $1.3$ nats of degradation (E1), and retraining 26.2M or 28.7M target-side parameters reduces it to about $0.25$ (E2, E4). More trainable capacity mitigates the damage but never turns it into a gain. HEMI, under the same budget and with the smallest trainable pack, gains $+0.172$, $+0.082$ and $+0.095$. The comparison therefore isolates the injection rule rather than capacity: E2 and E4 have about $400\times$ the target-side parameters of HEMI's alignment pack (26.2M and 28.7M against 65,811) and still fail, which agrees with Appendix C.1.3, where the same rendered vector added wholesale with a single learned scalar gives no measurable gain, and with the

Table 17: Attribution of the 14B gain to the content of the table, under the protocol of Table 16 (baseline PubMed 2.224, Math 2.261). All entries are single-seed (42) points, so the trained row is directly comparable with the controls; the three-seed mean of the trained row is reported in Table 16. The trained Math entry is from Table 15, the control protocol not having been run on Math.

| Table content | PubMed | Math |
|---|---|---|
| zero (content $\approx 0$) | $+0.127$ | $+0.045$ |
| random (no domain knowledge) | $-0.009$ | $-0.255$ |
| trained, rows shuffled | $+0.010$ | $-0.167$ |
| trained (HEMI) | $+0.022$ | $-0.149$ |

Table 18: The Engram injection rule against the 7B target, three domains, seed 42; $\Delta L$ positive is a reduction in validation loss. All rows share one reduced budget (1k samples, 2 epochs, $V = 20$k, layer 10); a larger budget might reverse these signs.

| Variant | PubMed | Math | Code |
|---|---|---|---|
| E1 Engram natively (1.5B→1.5B, no transfer), 7.5M trained | $-1.296$ | $-1.404$ | $-1.498$ |
| E2 frozen table, projection retrained on 7B, 26.2M trained | $-0.254$ | $-0.440$ | $-0.441$ |
| E3 frozen table, Procrustes rendering, nothing trained | $-11.46$ | $-11.51$ | $-12.24$ |
| E4 table and projection trained from scratch on 7B, 28.7M trained | $-0.256$ | $-0.434$ | $-0.403$ |
| HEMI, same budget, 65.8K trained | $+0.172$ | $+0.082$ | $+0.095$ |

gate and adapter ablations, the two components whose removal changes the sign of the gain. Two caveats bound the reading. First, the budget is small for every trainable variant, so E1, E2 and E4 may be under-trained and a larger budget might reverse their signs; we did not verify this. Second, because E1 gives no gain even on the backbone the table was trained on, these rows do not isolate transfer as the cause of the Engram variants' failure: what they support is the injection rule (C2) and the absence of a mechanism to control the magnitude of a wholesale injection, not a claim that transferring an Engram table is what breaks it.

### C.6 Numerical Verification of the Migration Stability Theorem

Theorem 1 bounds the direction error by $\frac{1}{2}\varepsilon_k^2$ with $\varepsilon_k = \nu_f/\sqrt{f_k} + \nu_{\text{drift}} G_{\max} + \delta_{\max}$; expanding the square (Appendix A) gives a mismatch term $\frac{1}{2}\delta_{\max}^2$ (C3), a drift floor $\frac{1}{2}\nu_{\text{drift}}^2 G_{\max}^2$ (C2), a finite-frequency term $\nu_f^2/(2f_k)$ (C1) and three cross terms, of which $\nu_f \nu_{\text{drift}} G_{\max}/\sqrt{f_k}$ couples drift and residual. This appendix checks the scaling of the three squared terms and the validity of the full bound on synthetic constructions in which the quantities of Appendix A are set directly, so that one source is excited at a time; it tests the algebra of the bound, not the modeling assumptions behind Lemmas 1, 2 and 4, which the end-to-end experiments address only indirectly. We compare the fitted power-law exponents of the direction error $1 - \cos(p_S, \Pi p_B)$ against the theoretical exponents. The construction uses the deterministic instantiation of Appendix A: memory vector $v_k = s_k + \varepsilon_k$ with $\|\varepsilon\|/\|s\| = \sqrt{n_{\text{ctx}}/f_k}$, that is $\nu_f = 2\sqrt{n_{\text{ctx}}}$ at $c_0 = f_k/f^*$; $\Phi_S$ and $\Phi_B$ orthonormal renderings in a common space with maximum principal angle $\theta_{\max}$ (hence $\delta_{\max} = 2\sin(\theta_{\max}/2)$); and a Stage-2 drift chord $\nu_{\text{drift}} G_{\max} = 2\sqrt{2}\, G_{\max}$. Three paired settings each excite one term: the finite-frequency term compares the noisy rendering with the rendering of the signal direction, the mismatch term compares the two models' renderings of the signal direction, and the drift term compares the target's undrifted rendering with the drifted direction.

**Reading.** (a) V1 approaches $-1$ for $f_k \geq f^*$ ($-0.78$ over the full range), so the residual term scales as $O(1/f_k)$ in the regime the theorem covers; points below $f^*$ deviate from the power law, which is the boundary of that regime. (b) V2 and V3 give exponents near 2 for the drift floor and the mismatch term, as the bound states. (c) The bound holds at all 120 grid points and is loose (ratio $\leq 0.12$), so the sum of the terms is a safe upper bound rather than a tight prediction. (d) The fitted coefficient of V1 is $\approx 71/\sqrt{n_{\text{ctx}}}$, and the direction error depends on $f_k$ only through $f_k/f^*$ in these constructions.

Table 19: Direct numerical verification of the migration stability theorem: fitted versus theoretical power-law exponents.

| Verification item | Fitted exponent | Theoretical exponent | Match? |
|---|---|---|---|
| (V1) direction error vs. $f_k$ residual term | $-0.93$ (asymptotic region $f_k \geq f^*$) | $-1 (\propto 1/\, f_k)$ | ✓ |
| (V2) direction error vs. $G_{\max}$ drift term | 1.83 | $2 (\propto G_{\max}^2)$ | ✓ |
| (V3) direction error vs. $\delta_{\max}$ mismatch term | 2.00 | $2 (\propto \delta_{\max}^2/2)$ | ✓ |
| (V4) bound validity | 120/120 grid points satisfy $1 - \cos \leq$ RHS | max ratio $= 0.12$ | ✓ |

# D Additional Discussion

## D.1 Boundaries with the Closest Prior Work

Table 20 states, for the lines of work closest to HEMI, what is shared, what HEMI changes, and why the two are not interchangeable. The entries describe the source of each difference, not scores.

## D.2 Relation Between Differential Injection and Knowledge Editing

Differential injection can be read as a soft, batched form of knowledge editing. Editing methods such as ROME (Meng et al., 2022), MEMIT (Meng et al., 2023) and MEMOIR (Wang et al., 2025) modify specific weights to change one fact at a time, with high precision per edit and a cost that grows with the number of edits. Differential injection instead stores many n-gram contexts in one table, routes each to the tokens where it applies through the hash, and lets the gate decide the injection strength; it is less precise than an edit but batches naturally and, through the rendering matrix, is not tied to one backbone.

# E Baseline Implementations and Hyper-parameters

All baselines share the data, the development/test split of Section 4.1 and the evaluation code. Hyper-parameters are chosen on the development part from the grids given below; where a selected value differs between domains, the values are listed in the order PubMed, Math, Code, General.

## E.1 Data

The four corpora are built from public sources with one pipeline. General is the English split of C4 (Raffel et al., 2020); Code is CodeParrot-clean (Hugging Face, 2022), deduplicated Python files from GitHub, restricted to permissively licensed files; Math is OpenWebMath (Paster et al., 2024); PubMed is the 2026 PubMed baseline of the National Library of Medicine (National Library of Medicine, 2026), keeping English, non-retracted records that have an abstract. Documents are filtered for length and encoding, near-deduplicated with MinHash LSH (64 permutations, threshold 0.8, 5-token shingles), and screened against the test sets of HumanEval, MBPP, GSM8K, MATH and PubMedQA by exact and 24-token fragment matching, with flagged documents removed. Each domain provides 300M training, 10M validation and 10M test tokens, assigned by a hash of the document group (repository for Code, host for web text, PubMed identifier otherwise) so that a group never straddles two parts; the samples of Section 4.1 are drawn from these files.

## E.2 Baseline Groups

*Non-parametric:* retrieval-augmented generation in the in-context form of Ram et al. (2023), with a BM25 index over the domain training set. *Target-side fine-tuning:* LoRA (Hu et al., 2022) with $r = 64$ and $r = 8$ on the attention and MLP projections (161.5M and 20.2M parameters); LoRA with $r = 1$ on the query projection (201K); and prompt tuning (Lester et al., 2021) with 18 soft tokens (65K), the last two matching the budget of $\theta$. *Trained once on the 1.5B model and reused:* a

Table 20: Boundaries with the closest prior work.

| Work | Common ground | What HEMI changes | Why not interchangeable |
|---|---|---|---|
| Engram (Cheng et al., 2026) | Memory stored as directions in an n-gram hash table and injected at one layer | Rendering through an orthonormal anchor basis instead of a trained projection; injection of the gated difference instead of the gated value; table size from the critical frequency instead of search | Engram's projections and gate are trained with the table and shaped by $d_h$, so the table is tied to one backbone; HEMI recomputes $\Phi_B$ from the target's embeddings and retrains only $\theta$ |
| Tokenizer-agnostic Engram (Lim & Chieu, 2026) | A memory table reused across configurations | They make the hash compatible across tokenizers; we make the stored directions readable across backbones that share a vocabulary | Orthogonal problems: with vocabularies aligned, hidden-space geometry still differs across backbones, which is what anchor rendering addresses |
| LoRA, Adapters, prompt tuning (Hu et al., 2022; Houlsby et al., 2019; Lester et al., 2021) | The backbone is frozen and few parameters are trained | Their parameters are defined relative to one backbone's weights or inputs; HEMI's $\theta$ lives in the $d_m$-dimensional memory space and the knowledge is in a table other backbones can read | A LoRA or prompt trained on one backbone does not carry over without a transfer step; a HEMI table does, with $\theta$ retrained. At a matched budget the two are within a few hundredths on 7B (Section 4.2) |
| ReFT (Wu et al., 2024) | A hidden state is projected into a low-dimensional subspace and a difference is added back ($h + R^\top(Wh + b - Rh)$ in LoReFT) | The target value comes from a table addressed by the n-gram hash, not from a linear function of $h$; the subspace is fixed by anchor tokens, not learned; the table is reusable across backbones | ReFT's $R$ and $W$ are trained per backbone and carry no external memory; HEMI's intervention is a memory read whose content was learned on another backbone |
| RAG, kNN-LM (Lewis et al., 2020; Khandelwal et al., 2020) | External knowledge enters at inference | RAG concatenates text into the context and kNN-LM interpolates output distributions; HEMI writes knowledge into a table in Stage 1 and injects it into a hidden state, with no context cost and no datastore search | Different cost model: retrieval per window or per token against one hash lookup; on our domains in-context retrieval changes the loss by at most 0.03 (Section 4.2) |
| Memory Decoder, proxy-tuning, CombLM (Cao et al., 2025; Liu et al., 2024; Ormazabal et al., 2023) | A domain component is trained once on a small model and reused on larger models of the family | They combine output distributions and run the small model at every token; HEMI injects inside the target's forward pass through a table lookup | Their gains are higher in some domains (Section 4.2); HEMI's structural difference is zero extra forward passes at inference |

modified Kneser-Ney 3-gram model interpolated at the output layer; Memory Decoder (Cao et al., 2025), a small decoder whose output distribution is interpolated with the target's; proxy-tuning (Liu et al., 2024; Mitchell et al., 2024) and CombLM (Ormazabal et al., 2023), which reuse a domain-adapted 1.5B model through logit differences or probability interpolation; and HEMI.

### E.3 Training Hyper-parameters

All baselines are run with seed 42 and trained on 100M tokens of the domain training set; HEMI is trained on the 12,000 samples per domain (5,000 for Code) of Section 4.1. Adapters and small models are trained with AdamW without weight decay, a linear warm-up over the first 5% of the

steps followed by cosine decay to zero, gradient clipping at 1.0, sequences of 2,048 tokens and 128 sequences (262K tokens) per step, in bfloat16 with the trainable parameters kept in fp32. LoRA scales its update by $\alpha_{\mathrm{LoRA}}/r$ with $\alpha_{\mathrm{LoRA}} = 16$ and zero-initialises $B$, so that training starts from the base model. Target-side adapters are trained on 100M tokens of the domain training set. Learning rates are $10^{-3}$ for LoRA $r = 8$ and $r = 64$ and $3 \times 10^{-3}$ for LoRA $r = 1$ and prompt tuning, except $3 \times 10^{-3}$ for $r = 64$ on Math and $10^{-3}$ for $r = 1$ and prompt tuning on PubMed; where a grid was run, the value was chosen from $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$ by 20M-token probes on the development part and the adapter was then retrained from scratch at the selected value. The domain adaptation of the 1.5B model that proxy-tuning and CombLM reuse is LoRA with $r = 64$ on all projections, trained on 100M tokens at $3 \times 10^{-3}$ (Code, General) or $10^{-3}$ (PubMed, Math). EMI's own optimizer, learning rates and training length in Stage 1 and Stage 2: AdamW without weight decay, gradient clipping at 1.0, bfloat16 with the trainable parameters kept in fp32, and a linear warm-up over the first 5% of the steps followed by cosine decay to zero, at 128 sequences of $2{,}048$ tokens per step. Stage 1 trains the table and $\theta$ on the source backbone on the $12{,}000$ samples per domain ($5{,}000$ for Code) of Section 4.1; Stage 2 retrains only $\theta$ on the target on the same budget. Learning rates are selected on the development part as described above for the adapters; the Stage-1 table learning rate $\eta_V$ takes the values $2 \times 10^{-4}$ and $2 \times 10^{-3}$ in the sweep of Appendix C.2, and the learning rate of $\theta$ is selected from $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$.

### E.4 Output-Interpolation Baselines

*KN 3-gram:* interpolated modified Kneser-Ney with discounts from count-of-counts, trained on the domain training set; $p = (1 - \lambda)p_{\mathrm{LM}} + \lambda p_{\mathrm{KN}}$ with $\lambda$ chosen on the development part. The counts are taken over the first 100M tokens of the training set; $\lambda \in \{0, 0.02, 0.05, 0.1, 0.15, 0.2, 0.3\}$, selected as 0.05, 0.05, 0.02, 0.1. *Memory Decoder:* a simplified form of Cao et al. (2025) in which the memory is a Qwen3-0.6B model (Qwen Team, 2025) fully fine-tuned on 100M tokens of the domain training set (learning rate $10^{-4}$ from $\{10^{-5}, 3 \times 10^{-5}, 10^{-4}\}$); the DeepSeek-R1-Distill-Qwen series has no member of this size. Its output distribution is interpolated with the target's, $p = (1 - \lambda)p_{\mathrm{LM}} + \lambda p_{\mathrm{MD}}$, with $\lambda \in \{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$, selected as 0.5, 0.4, 0.4, 0.5. *Proxy-tuning:* the 1.5B model is adapted with LoRA $r = 64$ (Appendix E.3); the target's logits are shifted by $\alpha(z_{\mathrm{tuned}} - z_{\mathrm{base}})$ with $\alpha \in \{0.5, 1, 1.5, 2\}$ selected on the development part; $\alpha = 0.5$ is selected in all four domains, and the default $\alpha = 1$ of Liu et al. (2024) is worse in every domain. *CombLM:* the same adapted 1.5B model is combined by probability interpolation with $\lambda \in \{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$; $\lambda = 0.5$, the largest value of the grid, is selected in all four domains.

### E.5 Retrieval-Augmented Generation

In-context RALM (Ram et al., 2023): a BM25 index ($k_1 = 1.2$, $b = 0.75$) over the first 100M tokens of the domain training set, split into passages of 256 tokens. Every 64 tokens the preceding 32 tokens form the query, the top-1 passage is prepended to the context, and the loss is computed on the original tokens only.

### E.6 Budget-Matched Adapters

LoRA with $r = 1$ on the query projections of all layers (201K parameters) and prompt tuning with 18 soft tokens initialized from random vocabulary rows ($18 \times d_h \approx 65$K), both trained with the settings of Appendix E.3.