THE CIRCUIT BASIS OF METACOGNITIVE FAILURE IN LANGUAGE MODELS: CONFLICT AND HALLUCINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models often answer with high confidence, as measured by the entropy of the output distribution, even when they are wrong. We study two situations that lead to such errors, *knowledge conflict*, where the context contradicts a fact stored in the weights, and *missing knowledge*, where the queried fact was never learned and the model *hallucinates*. To control exactly what the model has learned, we fine-tune it on synthetic facts that map arbitrary entity identifiers to random codes, installed through low-rank adapters (LoRA) on the multi-layer perceptron (MLP), query–key (QK), or value–output (VO) components. Our analysis suggests that MLP is the main substrate for storing facts in *basins*, regions of representation space into which a fact's representations converge, while QK steers generation toward a basin or the context, and VO writes the retrieved content into the residual stream. *Under knowledge conflict*, the arbitration between the stored fact and the context fails in a way that depends on how the fact is stored. A fact learned from a single phrasing yields to the context, but errors accumulate in the later digits, while a fact learned from many phrasings overrides the context, and the answer drifts into one that matches neither source. *Under missing knowledge*, the hidden state reaches no basin, yet every adapter placement produces confident hallucinations, most of all through VO. In both situations, the output entropy reflects how sharply the model commits rather than whether the right fact was retrieved, even though this information remains present in its representations. Across 12 pretrained models, the overall hallucination rate falls with scale, but a growing fraction of the remaining hallucinations are delivered with high confidence. Our mechanistic insights thus pinpoint a systemic metacognitive gap that does not close with scale in the models we study.

1 INTRODUCTION

Language models often answer with high confidence even when they are wrong. We study two situations that lead to such errors. In *knowledge conflict*, the context contradicts a fact stored in the model's weights, and the model may answer from either source, or from neither (Longpre et al., 2021; Neeman et al., 2023). In *missing knowledge*, the model is asked about something it never learned and answers fluently anyway, which is a *hallucination* (Ji et al., 2023; Zhang et al., 2025). In both situations, the output looks the same as it does when the model correctly recalls a fact. These errors are usually attributed to miscalibration, which post-hoc recalibration such as temperature scaling could in principle fix (Guo et al., 2017). This explanation assumes that the model's confidence carries the right information on the wrong scale. Nonetheless, it has not been systematically studied where the information that separates a correct answer from an error lives in the model, and why the confidence does not reflect it.

Studying this requires knowing exactly what a model does and does not know, which natural language does not allow, since a fact may have been seen during pretraining and a correct answer could come from recall, inference, or a lucky guess. Here, we instead teach a pretrained model 1,600 associations between arbitrary entity identifiers and random five-digit codes, which it cannot infer or recover from pretraining. Our design has three degrees of freedom, each isolating one aspect of the problem: i) *what* the model knows, which we control exactly; ii) *how robustly* it knows it, set by training on a single phrasing or on many different variations; and iii) *where* the knowledge is installed, set by placing low-rank adapters (LoRA) (Hu et al., 2022) on QK, VO, or MLP. We describe the results in terms of the generation trajectory, i.e. the path the hidden state follows through representation space

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

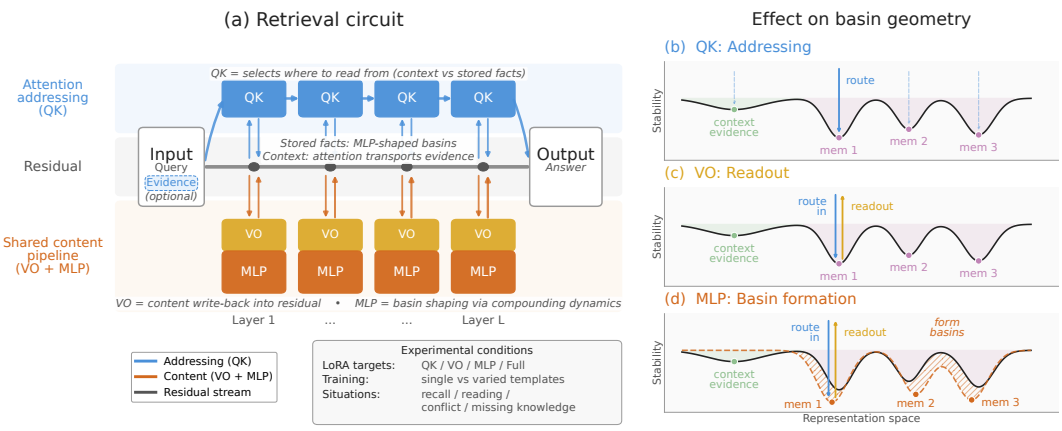

Figure 1: **The retrieval circuit and its basin geometry.** *(a)* The circuit behind contribution 1: an addressing mechanism (QK, blue) that steers the trajectory toward the context or toward stored content, and a content pipeline (VO and MLP, orange) that writes content into the residual stream and shapes the state. *(b–d)* The effect of placing an adapter on each component, behind contributions 2 and 3. QK changes which basin is selected without reshaping any, VO changes what is read out within the selected basin, and MLP reshapes the basins themselves. The output entropy depends only on how sharply the final state commits, and does not read whether a basin was reached or which source won.

during generation. When a fact is learned under many phrasings, these paths converge into a common region, which we call a *basin* (definition to come in Section 3.2).

Contributions. Under this setting and framework, we make the following observations. The first establishes the roles of the circuit components, the next two show how these roles produce errors in each situation, and the last shows how the problem changes with scale:

1. **QK, VO, and MLP play distinct roles in retrieval (Section 3).** MLP is the main substrate for storing facts in basins, consistent with its known role as a key–value memory (Geva et al., 2021; Meng et al., 2022). QK is consistent with steering the trajectory toward a basin or toward the context, and VO with writing the retrieved content into the residual stream. We infer these roles from the Jacobians of the pretrained model and support them with adapter placement.
2. **Under knowledge conflict, the arbitration fails in a way that depends on how the fact is stored (Section 4.1).** A fact learned from a single phrasing has a shallow basin, so the context wins, but MLP and Full adapters make late errors that compound autoregressively. A fact learned from many phrasings has a basin deep enough that the model can no longer read the context at all, which we call *context deafness*, and under a direct contradiction the answer starts on the stored code and drifts into one that matches neither source. In both cases, entropy stays low.
3. **Under missing knowledge, every adapter placement produces confident hallucinations (Section 4.2).** Without an adapter, the base model declines to answer about unseen entities. With any adapter, it produces a confident code even though its hidden state lies far from every basin. VO makes these hallucinations the most confident, while MLP and Full bias them toward memorized codes.
4. **The share of confident hallucinations grows with scale (Section 5.3).** Across 12 pretrained models from four families, larger models hallucinate less often, but a larger share of their remaining hallucinations are confident. We observe an empirical law that predicts this trend from the output logits.

An important consequence of our observations is what they reveal about the model's lack of metacognition, that is, its inability to know what it knows. The model's confidence reflects how sharply the final hidden state commits to one token, regardless of whether the right fact was retrieved. Under

missing knowledge, the hidden state still separates a correct answer from a hallucination, since its distance to the nearest stored fact differs sixfold between the two, and this information survives to the output logits, yet the entropy does not reflect it (Section 5). Across the models we study, from 0.36B to 14B parameters, this mismatch grows with scale rather than shrinking.

2 Experimental Design

Dataset. We generate 1,600 associations of the form `E000137` → `48210`. Identifiers are arbitrary strings and codes are five digits drawn uniformly at random, so that nothing about an entity predicts its code, and the model cannot infer these facts or recover them from pretraining. The model has either memorized a given mapping or it has not, with nothing in between. We then ask *"What is `E000137`'s code?"* in four situations.

	the fact is in the weights	the code is in the prompt
recall	yes	no
reading	no	yes
knowledge conflict	yes	yes, and it disagrees
missing knowledge	no	no

In recall and reading, the correct answer is available to the model. Knowledge conflict and missing knowledge are the two situations that lead to confident errors. In knowledge conflict, the code in the prompt is the stored code with its first digit changed, so that the two sources disagree at the first digit and agree at the remaining four.

Models. We fine-tune Qwen2.5-3B-Instruct (36 layers, width 2048) with rank-8 LoRA adapters for 10 epochs. Placing the adapter on QK, VO, MLP, or all three (Full) installs the same knowledge through different parts of the circuit. The adapters share every other hyperparameter, so any difference in behavior is attributable to where the knowledge was installed (details in Appendix A).

Training regimes. We train the models in two ways. Under *single-template* training, every example uses the same phrasing, ``Q: What is {entity}'s code? A:'', repeated 100 times per entity. This produces *brittle* storage, in which recall is bound to the training phrasing. Under *varied-template* training, 53 phrasings span direct questions, embedded statements, conversational turns, and distractor prefixes. This produces *robust* storage, in which recall survives rephrasing. As we show in Section 3.2, this choice determines whether a fact forms a basin at all, and it therefore shapes most of our results. Since neither training set contains any example of conflict, refusal, or reading a code from context, the model's behavior in those situations is never explicitly taught.

Evaluation. We use greedy decoding and record **accuracy**, whether the generated string contains the target code, along with the error at each digit position and the rank of the correct token at the first digit. Evaluation set sizes differ across analyses, from 30 to 1,600 queries, and are listed in Appendix Table 3. We measure the model's **confidence** by the entropy of the next-token distribution restricted to the ten digit tokens at the first digit of the answer, which is what an output-based monitor would see. Section 3 adds a third measure, read from the hidden state rather than the output.

3 Circuit Roles and Basin Geometry

To attribute an error to a specific component, we first need to know what each component does during recall. In this section, we identify the role of each component from the pretrained model, following the circuits framework (Elhage et al., 2021), define basins and verify that they are installed by training, and give predictions for how each role should shape the errors that we measure in Section 4.

3.1 Jacobian Decomposition of Component Roles

With the weights fixed, each sublayer applies a residual update $h \mapsto h + f(h)$, which views the transformer as a dynamical system over depth (Geshkovski et al., 2025). We extend this view to generation, where the hidden state follows a trajectory under an input-conditioned discrete-time system,

$$h_{t+1} = F(h_t;\, x,\, y_{\le t}), \tag{1}$$

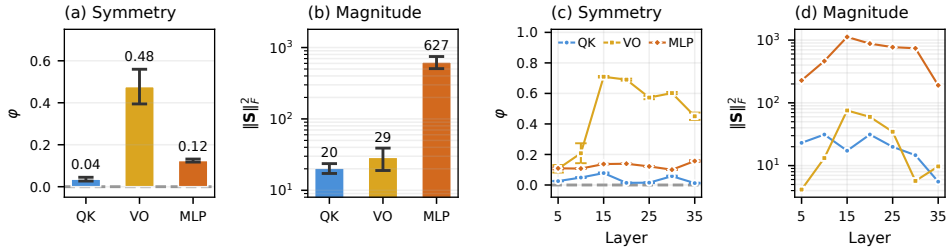

Figure 2: **Jacobian decomposition of the pretrained model.** Exact Jacobians at seven layers, averaged over five prompts. *(a)* Symmetry correlation φ. QK is unstructured, and VO is strongly symmetric. *(b)* Contracting magnitude $\|S\|_F^2$. MLP dominates by $\sim$25$\times$. *(c,d)* The same quantities across layers.

where F composes the attention, feed-forward, residual, and normalization maps. The local effect of a sublayer is given by its Jacobian J, which we split into symmetric and antisymmetric parts,

$$J = S + A, \qquad S = \tfrac{1}{2}(J + J^\top), \qquad A = \tfrac{1}{2}(J - J^\top). \tag{2}$$

The symmetric part S contracts or expands the hidden state, and the antisymmetric part A rotates it. We summarize each component by its symmetry correlation $\varphi = \mathrm{corr}(J_{ij}, J_{ji})$ over $i < j$, which ranges from -1 for a purely rotational map to $+1$ for a gradient-like one, and by $\|S\|_F^2$, the strength of its contraction.

In Fig. 2, we show both quantities for the pretrained model, computed from exact 2048×2048 Jacobians at seven layers. Each component has a distinct signature, which we use to assign its role. MLP has by far the largest symmetric part, about $25\times$ that of either attention component (Fig. 2b), and it dominates at every layer (Fig. 2d). It is therefore the main force pulling the hidden state toward particular regions, and we treat MLP as the main substrate for **storing** facts. QK has near-zero symmetry ($\varphi \approx 0.04$, Fig. 2a) and a small symmetric part. It does not reshape the hidden state but changes which token is attended to, and so **steers** the trajectory toward stored content or toward the context. VO is strongly symmetric ($\varphi \approx 0.5$ on average, peaking at 0.71 at layer 15, Fig. 2a,c), which is the signature of copying content along fixed directions into the residual stream, and we therefore treat VO as a **readout** of retrieved content, which is distinct from the output head that converts the final hidden state into token probabilities and sets the model's confidence. Since these signatures appear in the pretrained model before any fine-tuning, and the VO signature replicates across five model families (Appendix H), these signatures appear to be properties of the pretrained architecture rather than artifacts of our fine-tuning, while training determines what gets stored. Adapter placement supports this assignment. Installing the same facts through MLP requires about 3–4$\times$ fewer gradient steps than through QK (Appendix B), and only placements that include MLP produce recall that survives rephrasing (Fig. 4a), although placement shows what a component can learn when it carries the task, which constrains but does not uniquely determine its role in the pretrained model.

3.2 Basins and the Geometric Margin

To ask whether the hidden state reached a stored fact, we need a reference point for each fact. For each trained entity i, we average the final-layer hidden state over three canonical phrasings and call the result its *memorized state* m_i. For a query x with hidden state $h(x)$ at the first-digit position, we define the **margin** as the distance to the nearest memorized state and the **gap** as how much farther the second-nearest one is,

$$\delta(x) = \min_i \|h(x) - m_i\|_2, \qquad \mathrm{gap}(x) = \delta_2(x) - \delta(x). \tag{3}$$

A small margin means that the query reached the general vicinity of the memorized representations, and a large margin means that it reached nothing. We call the region around m_i a fact's *basin* (Fig. 3b), by analogy with a basin of attraction, a region that hidden states are pulled into and stay in. We use this term because the region has three measurable properties: i) *contraction*, since the MLP update pulls nearby hidden states together, with about three-quarters of its symmetric eigenvalues negative through layers 10–25; ii) *convergence*, since, as a result, hidden states from a fact's many phrasings end up close to its memorized state, so that recalled queries sit at a margin of about 22

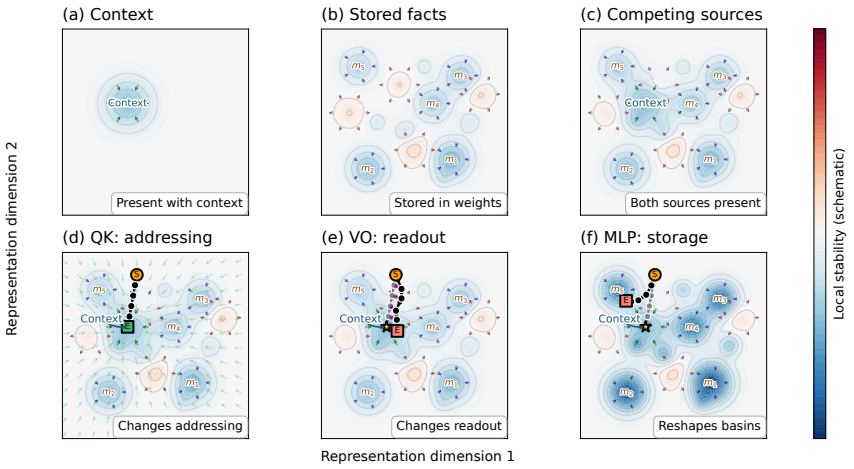

Figure 3: **Schematic basin geometry at a single layer.** The wells are basins in the finite-time sense of Section 3.2, regions the trajectory is pulled into at one step of generation, not a global landscape. *(a)* A code in context creates a transient basin, present only while those tokens are present. *(b)* Memorized facts create basins present for every input. *(c)* With both, the trajectory is pulled toward one or the other. *(d)* A QK adapter changes where the trajectory is steered, so it reaches the context. *(e)* A VO adapter changes what is read out, so the answer lands near, but not at, the target. *(f)* An MLP adapter deepens the memorized basins, which capture the trajectory before it reaches the context.

while never-learned queries sit at about 133, six times farther (Fig. 6a); and iii) *stability*, since the robust store keeps producing its answer under about twice as much embedding noise as the brittle store (Appendix Fig. 11). Unlike a basin of attraction, however, a basin here exists only over the finite number of layers and generation steps, since the composed map has no fixed points (Appendix E). The context can create a basin as well. A code given in the prompt pulls the hidden state toward it while those tokens are present (Fig. 3a), so under knowledge conflict the trajectory is pulled toward two basins at once (Fig. 3c).

Two controls show that basins are installed by training on varied phrasings rather than read off the entity strings. The margin does not separate recalled from never-learned queries in the base model, nor in brittle adapters that recall their own phrasing perfectly (Appendix E).

3.3 Predictions

Since an adapter changes only the contribution of the component it is placed on, the roles above give predictions for each placement (Fig. 3d–f): 1) QK should read the context faithfully but cannot store a fact robustly, since it steers without reshaping basins; 2) VO should reach roughly the right region but read it out imprecisely, with errors spread across digit positions; and 3) MLP should store facts robustly in deep basins that also capture trajectories that should follow the context, increasingly as the generated digits accumulate. For a never-learned entity, the hidden state should reach no basin under any placement, while the output head should commit to an answer regardless. We test these predictions in Section 4.

4 Failure Modes

In this section, we test these predictions in the two situations that lead to confident errors. We find that the component roles of Section 3 determine how the model fails in each. Under knowledge conflict (Section 4.1), how robustly a fact is stored decides whether the weights or the context win, and each adapter placement leaves its own error pattern. Under missing knowledge (Section 4.2), the hidden state reaches no basin, yet the output head still commits to an answer, and the placement determines how confident that answer is.

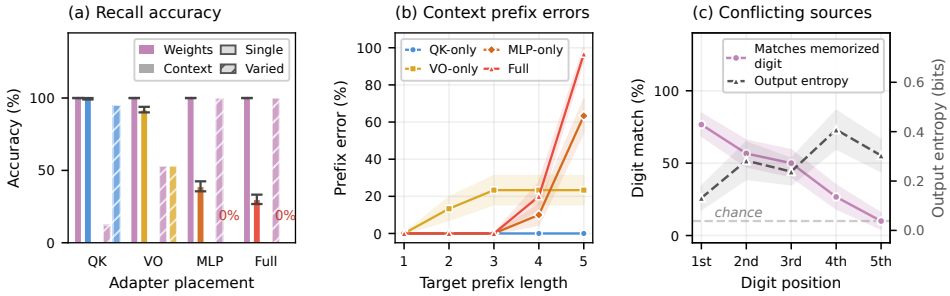

Figure 4: **Error patterns by adapter placement under knowledge conflict.** *(a)* Recall from the weights and from the context by placement, under single-template (brittle) and varied-template (robust) training. QK preserves context reading but cannot store facts robustly, while storing facts robustly through MLP or Full drives context reading to zero. *(b)* Prefix error, the fraction of answers whose first k digits do not match the code in the context, for brittle adapters (30 queries each). QK makes no errors, VO's errors appear by the third digit and do not grow further, and MLP and Full make late errors that compound. *(c)* Under a contradiction with a robustly stored fact, the answer begins with the stored code and drifts away from it, while output entropy stays low.

4.1 Knowledge Conflict

Under knowledge conflict, the correct answer is always stored in the weights. What fails is the arbitration between the weights and the context, and how it fails depends on how robustly the fact is stored (Fig. 4).

Brittle storage. When a fact is learned from a single phrasing, the context wins, but the placements that reshape the basins corrupt the end of the answer. Brittle adapters recall 100% of stored facts on the training phrasing but nearly 0% under any other phrasing, so their basins are inactive in a new prompt, and in a direct contradiction the stored code never wins (0 of 360 trials across adapters and instructions; Appendix Table 6). The placement determines how well the model reads the context instead (Fig. 4a), with QK at 99.5%, VO at 92%, and MLP and Full at 39% and 30%. It also determines where the errors appear (Fig. 4b). QK makes almost none. VO's errors appear by the third digit and stop growing, consistent with an imprecise readout at every step, as in prediction 2). MLP and Full read the first three digits correctly but fail at the fourth and fifth (63% and 97% error at the last digit), consistent with stored codes capturing the trajectory once enough digits have been generated, as in prediction 3) (sample outputs in Appendix D.3).

Robust storage. When a fact is learned from many phrasings, the stored fact wins and the model stops reading the context. MLP and Full recall 100% of stored facts across phrasings, but their context reading drops to zero, which we call *context deafness* (Fig. 4a). QK shows the opposite tradeoff, reading the context 95% of the time but recalling only 12% of stored facts, and VO falls in between at 53% on both. The component that stores facts robustly is therefore the same one that blocks the context, as in predictions 1) and 3).

Conflict under robust storage. When a robustly stored fact meets a contradicting context, the answer starts on the stored code and then drifts into one that matches neither source (Fig. 4c). Since the code in the prompt differs from the stored code only in its first digit, the two sources disagree at the first digit and agree at the rest. The model follows its weights at the first digit about three times in four, but even at the remaining digits, where the two sources agree, recall of the stored digit falls from 77% to 10%. Of thirty answers, none reproduces the prompt's code and only two reproduce the stored one, while the entropy stays between about 0.1 and 0.4 bits. Because this error unfolds across the answer, it is not visible at the first digit, where the query is still nearest its own memorized state in 96% of cases, and signals read there separate these errors poorly (AUROC below 0.75; Appendix D.4).

Hence, based on these results, we conclude that knowledge conflict is a failure of arbitration rather than of storage, and that the arbitration is set by how the fact is stored rather than by what the two sources say. Retrieval fails even at the digits where the two sources agree, and with brittle storage,

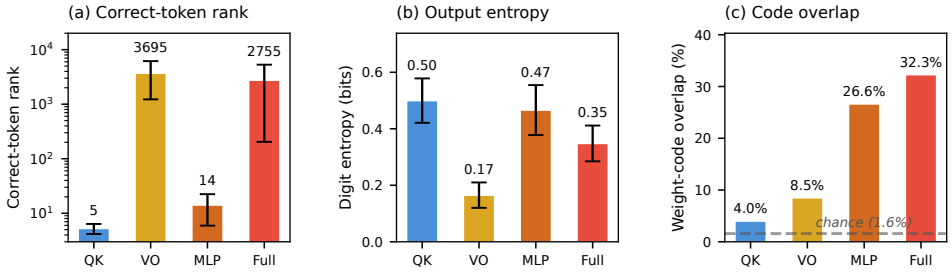

Figure 5: **Hallucination by adapter placement.** Never-learned entities. *(a)* Rank of the correct token. VO buries it beyond rank 3,600 (Full likewise), while QK and MLP keep it accessible (rank 5 and 14). *(b)* Digit entropy. VO is the most confident (0.17), and QK and MLP the least (0.50 and 0.47). *(c)* Fraction of hallucinated codes that match a memorized code. MLP and Full reproduce memorized codes 17–20× more often than chance (1.6%).

giving the same code in both does not reliably help and can even hurt (Appendix D.2). The generation trajectory is therefore largely fixed by the storage structure, while the entropy stays low.

4.2 Missing Knowledge and Hallucination

Under missing knowledge, there is no stored fact to retrieve. The hidden state reaches no basin, yet the output head still commits to an answer, and the adapter placement determines how confident that answer is (Fig. 5).

Basin absence. Without an adapter, the base model declines to answer queries about entities it has never seen, producing a code for only 3% of them (Appendix D.3). With any adapter, it instead produces a confident five-digit code (Appendix Table 7), even though the hidden state never reaches a basin. For the robust MLP adapter, it lands about six times farther from any memorized state than for a recalled fact (133 versus 22), and none of a hundred never-learned queries falls within the range of recalled queries. Our directional analysis suggests why the answer is confident anyway. The output head is sensitive only to particular projections of the hidden state. When a hallucination reproduces another entity's code, its hidden state lies about 175 units from that entity's memorized state, but along the direction that selects the emitted digit the two states nearly agree, and both assign that digit a high logit (about 31 and 42). Token-level confidence therefore does not distinguish reaching a basin from merely reproducing its code.

Effect of adapter placement. Every placement hallucinates, but the placement determines the character of the hallucination. VO produces the most confident hallucinations (entropy 0.17) and buries the correct token beyond rank 3,600. QK and MLP are the least confident (0.50 and 0.47) and keep the correct token near the top (rank 5 and 14). MLP and Full, which reshape the basins globally, bias the answers toward memorized codes, reproducing one 26.6% and 32.3% of the time, compared to 1.6% by chance. This bias arises at the output rather than from capture by a specific basin, since for the robust MLP adapter only 9 of 56 reproduced codes (16%) belong to the nearest stored entity.

Hence, we conclude that in our setting the model's output does not distinguish a learned fact from a never-learned one. Once any component is fine-tuned, the model stops declining and answers never-learned queries just as it answers learned ones, and the output head commits to an answer whether or not the hidden state reached a basin. In other words, the model answers as if it had no step that checks whether the fact is stored, which is the metacognitive gap we examine in Section 5.

5 Why Output Confidence Fails

Section 4 shows that the model stays confident under both knowledge conflict and missing knowledge. In this section, we ask why the output entropy does not flag these errors (Section 5.1), what can flag them instead (Section 5.2), and how the problem changes with scale (Section 5.3).

5.1 Why Entropy Misses the Errors

To understand why the entropy fails to flag these errors, we first test whether the information separating a correct answer from a hallucination is lost before the output. Specifically, we train linear probes to classify correct versus incorrect answers across all five situations, from the hidden state and from the output logits. The two probes perform almost equally well (AUROC 0.965 and 0.960), and so does the entropy (0.967). The difference appears on the confident answers. Since these are selected for low entropy, the entropy separates them poorly by construction (0.665), but the logit and hidden-state probes still reach 0.898 and 0.888. The model therefore encodes whether a confident answer is correct, even though the entropy does not. This is because entropy only measures how peaked the distribution over the ten digits is, which depends on how strongly the final hidden state points toward one digit, not on whether it reached a basin. As shown in Section 4.2, the directions that encode whether a fact was stored barely affect the digit logits, so catching every hallucination by entropy would also reject 99.5% of correct recalls (Fig. 6b). We note that the same reasoning applies to knowledge conflict. As the answer drifts, the output head commits to each digit as sharply as under clean recall (Fig. 4c), since the conflict changes which digit is chosen, not how sharply.

5.2 Detecting the Errors

We next compare three ways to detect hallucinations, i) the margin, ii) a head trained to flag errors, and iii) consistency across paraphrases. The margin separates recalled from never-learned queries almost perfectly (Fig. 6a,b), but only given the memorized states of all stored facts. With a fact's own memorized state removed, it falls to chance (AUROC 0.50), so it serves analysis rather than detection. A head trained end to end on the hidden states closes only part of the gap (Appendix I). For consistency, we ask the same question in eight different phrasings and measure how often the answers agree. For stored facts, the model gives the same code in nearly all phrasings (agreement 0.97), while for never-learned entities the answers vary much more (agreement 0.52). Counting the number of distinct answers separates the two with an AUROC of 0.96, without requiring any list of stored facts, and it generalizes to new facts where the margin does not. This approach is a paraphrase-based variant of self-consistency and semantic entropy (Wang et al., 2023; Kuhn et al., 2023). Since every phrasing of a stored fact converges into the same basin, the answers agree, whereas a never-learned entity has no basin to converge to, so each phrasing ends somewhere different.

5.3 Scaling of Confident Hallucination

Finally, we ask whether larger models would fix this problem. Since basins cannot be measured in natural-language models (Appendix K), we test one prediction of our account on real models, using the full next-token entropy at the first answer token. Since the entropy reflects how sharply the output head commits rather than whether a basin was reached, we expect the opposite. In the models we evaluate, larger models separate their top two logits by a wider gap (Appendix L), and the softmax turns a wider gap into a sharper distribution whether or not the answer sits in a basin. Larger models should therefore deliver a larger fraction of their hallucinations with high confidence.

To make this prediction quantitative, let $\bar{\Delta}$ be the mean gap between the top two logits on wrong answers, and let Δ^* be the gap at which the entropy falls below a threshold H_0. The fraction of confident hallucinations is then the fraction of wrong answers whose gap exceeds Δ^*, $C = \Pr[\Delta > \Delta^*]$. If these gaps are exponentially distributed, this gives

$$C = \exp\Big(-\frac{c}{\bar{\Delta}}\Big), \qquad c = \Delta^*(H_0, V), \tag{4}$$

with no free parameter once Δ^* is measured. The measured gaps are close to exponential (coefficient of variation 1.06–1.24, versus 1) but not exactly so, so we expect Eq. 4 to capture the trend rather than exact values. In Fig. 6c, we show that it does. Across 12 instruction-tuned models (0.36B–14B, four families) and two benchmarks, the 21 resulting points follow Eq. 4 with $r^2 = 0.88$.

In a teacher–student ablation where only the width changes, the margin separation improves from $1.2\times$ to $153\times$ as C rises from 52% to 99% (Appendix L), so the basins sharpen with scale while the entropy does not use them. We note that larger models also hallucinate less overall, but only slowly ($H \propto N^{-0.27}$ across 17 models; Appendix L). Since C rises from below 1% to 24% over the same range, the fraction of all questions answered with a confident hallucination, $H \cdot C$, rises from under 0.5% below 1B parameters to more than 10% at 14B.

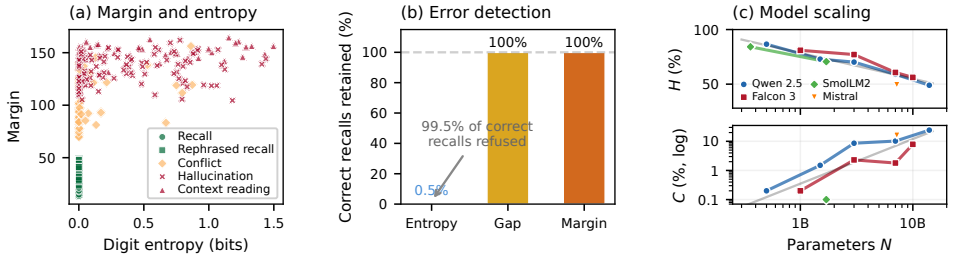

Figure 6: **Margin versus entropy, and confident hallucination across scale.** *(a)* Recall clusters at low margin and hallucination at high margin. *(b)* Catching every hallucination by entropy rejects 99.5% of correct recalls, and by margin none. *(c)* Across 12 models, hallucination rate H falls while confident fraction C rises.

6 Related Work

Prior work shows that models can be trained to predict whether they know an answer (Kadavath et al., 2022), that internal states encode correctness better than output probabilities (Azaria & Mitchell, 2023; Orgad et al., 2025), and that fine-tuning on new knowledge increases hallucination (Gekhman et al., 2024). These studies work in natural language, where what the model has learned is uncertain. We instead isolate this gap with exact ground truth and localize it to specific circuit components. Our storage claim builds on key–value accounts of feed-forward layers (Geva et al., 2021; Meng et al., 2022; 2023), and our conflict results relate to work on when weights or context win (Longpre et al., 2021; Neeman et al., 2023; Li et al., 2023; Xie et al., 2024). Calibration methods (Guo et al., 2017) inherit the blind spot of the output head, unlike hidden-state probes (Slobodkin et al., 2023; Kossen et al., 2024) and self-consistency (Wang et al., 2023; Kuhn et al., 2023).

7 Discussion

Key takeaway. Taken together, our results suggest that the model's confidence does not read the information it encodes about retrieval, namely whether a fact is stored (set by MLP basins) and which source the model follows (set by QK and the depth of storage). This information survives to the output logits, yet the entropy, which measures only how sharply the model commits to a token, does not use it. As a result, the model answers a contradicted or never-learned query as it answers a known one. Scale does not fix this. Larger models hallucinate less often, but since their outputs sharpen whether or not a fact was retrieved, a growing share of their remaining hallucinations are confident.

Implications. Our findings have two practical implications, one for how knowledge is added to a model and one for how its errors are detected. i) For adding knowledge by fine-tuning, where and how a fact is installed determines how the model later arbitrates between its weights and its context. Robust MLP storage gives reliable recall but can make the model ignore context meant to update or correct it, as in retrieval-augmented generation, and fine-tuning any component made the model answer questions that the base model declined. Evaluations of knowledge injection should therefore test context reading and unknown queries, not only recall. ii) For detecting errors, signals that read whether a fact was reached do better than output confidence, whose limit tightens with scale. The margin works when the stored facts are known and paraphrase consistency works otherwise, while conflict errors develop during generation and likely require monitoring the full answer.

Toward metacognition. For a model to know what it knows, its confidence would need to read this information directly, since better calibration of the entropy alone cannot recover it. Our results point to readouts trained on logits or hidden states, or aggregated across paraphrases or generated tokens.

Limitations. Our setting is synthetic and uses one model family, so generalization to naturally learned facts and other architectures remains open. An earlier natural-language extension was confounded by surface form (Appendix K), and the hallucination results rest on thirty entities per adapter.

AI USE STATEMENT

In this work, we used generative AI tools for drafting and editing portions of the manuscript text, for implementing and running analysis and evaluation code, and for making small, scripted edits to existing figure-generation code, such as removing a panel from a figure. We did not use generative AI tools to design the experiments, generate the synthetic dataset, or create the figures, which were produced by the authors. The circuit account of conflict and hallucination developed in this paper was formulated by the authors, based on interpreting the experimental results; generative AI tools were used in that process to help organize and discuss the evidence, but not to originate the central hypotheses or the methodology. We have reviewed all AI-assisted work, and code, analysis, text, and figure edits were checked by the authors before inclusion. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies factual recall and its failure modes in a fine-tuned language model, using a synthetic entity–code dataset and a publicly released base model (Qwen2.5-3B-Instruct). It does not involve human subjects, new data collection from people, or the release of new pretrained models, and we are not aware of ethical concerns specific to this work beyond those already associated with the cited base model.

REPRODUCIBILITY STATEMENT

Model, training, and evaluation details are given in Appendix A, including LoRA hyperparameters (Appendix A.3), prompt templates (Appendix A.1), and the evaluation protocol (Appendix A.5). The margin and gap used throughout are defined in Eq. 3, and the Jacobian decomposition behind the three circuit roles is defined in Eqs. 1–2. The scaling law and its per-model verification are given in Appendix L, and the end-to-end metacognitive head result referenced in Section 5 is detailed in Appendix I. Code and data will be released on acceptance.

REFERENCES

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning LLMs on new knowledge encourage hallucinations? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pp. 7765–7784, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.444. URL https://aclanthology.org/2024.emnlp-main.444/.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. A mathematical perspective on transformers. *Bulletin of the American Mathematical Society*, 62(3):427–479, 2025. doi: 10.1090/bull/1863.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 5484–5495. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.446. URL https://doi.org/10.18653/v1/2021.emnlp-main.446.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pp. 1321–1330. PMLR, 2017. URL http://proceedings.mlr.press/v70/guo17a.html.

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 Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pp. 2790–2799. PMLR, 2019. URL http://proceedings.mlr.press/v97/houlsby19a.html.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38, 2023. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1147. URL https://doi.org/10.18653/v1/P17-1147.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *CoRR*, abs/2207.05221, 2022. URL https://arxiv.org/abs/2207.05221.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth A. Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *CoRR*, abs/2406.15927, 2024. doi: 10.48550/ARXIV.2406.15927. URL https://doi.org/10.48550/arXiv.2406.15927. ICML 2024 Workshop on Foundation Models in the Wild.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612. URL https://aclanthology.org/P19-1612/.

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 Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix X. Yu, and Sanjiv Kumar. Large language models with controllable working memory. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Findings of ACL, pp. 1774–1793. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.112. URL https://doi.org/10.18653/v1/2023.findings-acl.112.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.353. URL https://doi.org/10.18653/v1/2021.acl-long.353.

Yizhou Liu, Ziming Liu, Cengiz Pehlevan, and Jeff Gore. Universal one-third time scaling in learning peaked distributions. In *Forty-third International Conference on Machine Learning, ICML 2026*, 2026. URL https://arxiv.org/abs/2602.03685.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 7052–7063. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.565. URL https://doi.org/10.18653/v1/2021.emnlp-main.565.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. Disentqa: Disentangling parametric and contextual knowledge with counterfactual question answering. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 10056–10070. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.559. URL https://doi.org/10.18653/v1/2023.acl-long.559.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson

Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. Llms know more than they show: On the intrinsic representation of LLM hallucinations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=KRnsX5Em3W.

Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. The curious case of hallucinatory (un)answerability: Finding truths in the hidden states of over-confident large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 3607–3625. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.220. URL https://doi.org/10.18653/v1/2023.emnlp-main.220.

Shangwen Sun, Alfredo Canziani, Yann LeCun, and Jiachen Zhu. The spike, the sparse and the sink: Anatomy of massive activations and attention sinks. *CoRR*, abs/2603.05498, 2026. doi: 10.48550/ARXIV.2603.05498. URL https://doi.org/10.48550/arXiv.2603.05498.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=auKAUJZMO6.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the AI ocean: A survey on hallucination in large language models. *Computational Linguistics*, 51(4):1373–1418, 2025. doi: 10.1162/coli.a.16.

APPENDIX

Throughout this appendix, "basin" is used in the finite-time, measured sense of Section 3.2, not as a global energy minimum.

A TRAINING DETAILS

A.1 PROMPT TEMPLATES FOR ROBUST STORAGE

Robust storage training uses 53 prompt templates organized into six categories, designed to cover a wide range of natural prompt structures. Each training example randomly samples one template; additionally, 25% of examples undergo synonym paraphrasing (substituting "code" $\rightarrow$ "identifier"/"number"/"value"/"designation", etc.) and 10% receive a distractor prefix (e.g., "Note: The system is operational."). Table 1 lists all 53 templates.

Category	Templates
A. Direct queries (18)	`Q: What is {entity}'s code? A:` · `Q: What is the code for {entity}? A:` · `What is {entity}'s code?` · `What is the code for {entity}?` · `Entity {entity}'s code is?` · `Give the code associated with {entity}.` · `Return the code for entity {entity}.` · `Provide the code for {entity}.` · `Tell me {entity}'s code.` · `What code corresponds to {entity}?` · `The code for {entity} is:` · `{entity} maps to what code?` · `Which code is assigned to {entity}?` · `What is the identifier for {entity}?` · `What is the mapping for {entity}?` · `What is the designation for {entity}?` · `{entity}'s identifier:` · `{entity}'s mapping:`
B. Embedded statements (10)	`Given the list below, what is the code for {entity}?` · `The records show: {entity} → ?` · `Find the appropriate code for {entity} from the database.` · `Look up the code for {entity} in the system.` · `According to our records, {entity}'s code is:` · `In the registry, {entity} corresponds to:` · `The system maps {entity} to:` · `Database entry for {entity}:` · `Retrieve the code associated with {entity}.` · `Query result for {entity}:`
C. Conversational (7)	`User: I need the code for {entity}. Assistant:` · `User: What's {entity}'s code? Assistant:` · `User: Can you look up {entity} for me? Assistant:` · `User: Tell me about {entity}. Assistant: {entity}'s code is` · `In our system, how would you encode {entity}?` · `If I asked for {entity}, what code would you give me?` · `Someone is asking about {entity}. What code should I tell them?`
D. Instructional (6)	`Provide the code that corresponds to {entity}.` · `List the code for {entity}.` · `Output {entity}'s code.` · `Display the code for {entity}.` · `Show me {entity}'s code.` · `Print the code for {entity}.`
E. Completion-style (5)	`{entity}'s code is` · `{entity} →` · `{entity} maps to` · `Code for {entity}:` · `The code is`
F. Distractor-laden (7)	`The weather is nice today. What is {entity}'s code?` · `Bob read about something yesterday. Also, the code for {entity} is:` · `Alice owns a car. The code for {entity} is:` · `Random note: the sky is blue. Now, what is the code for {entity}?` · `First, note that 2+2=4. Second, {entity}'s code is:` · `Consider the following: {entity} has the code:` · `Unrelated: cats are animals. Relevant: {entity}'s code?`

Table 1: All 53 prompt templates used for robust storage training. Each training example randomly samples one template and optionally applies synonym paraphrasing (25% probability) and distractor prefix injection (10% probability). The training data contains zero conflict examples, zero refusal examples, and zero context-reading examples.

A.2 DATA GENERATION

Entity–code assignments. Each entity is assigned an identifier of the form `E{i:06d}` (e.g., `E000000`–`E001599`) and a random 5-digit code sampled uniformly from $\{10000, \ldots, 99999\}$. With $N = 1{,}600$ entities, we obtain 1,586 unique codes out of 90,000 possible strings (1.76% coverage), so chance overlap between codes is low.

Brittle storage data. A single template ("`Q: What is {entity}'s code? A:`") is used for all training examples. Each entity is repeated 100 times, yielding $N \times 100$ training examples (e.g., 160,000 for $N = 1{,}600$). Examples are shuffled before training.

Robust storage data. The same $N = 1{,}600$ entity–code mappings are used with all 53 templates. Each of the 100 repetitions per entity randomly samples a template from the full set. With synonym paraphrasing (25%) and distractor prefixes (10%), this yields 160,000 examples, split 95/5 into 152,000 training and 8,000 validation examples.

Held-out evaluation sets. Five evaluation sets are constructed, each with 50 samples: (i) format generalization (novel templates never seen in training), (ii) distractor robustness, (iii) knowledge conflict, (iv) unknown/unseen entities, and (v) paraphrase variants.

A.3 LoRA HYPERPARAMETERS

Table 2 summarizes the LoRA configuration for all experiments. All adapters use the same optimizer settings; only target modules and rank differ.

Parameter	Value
Base model	Qwen2.5-3B-Instruct ($d = 2{,}048$, 36 layers)
LoRA rank r	8 (main experiments); 32, 64 (rank sweep)
LoRA α	$2r$ (i.e., 16, 64, 128)
LoRA dropout	0.0
Learning rate	5×10^{-5}
LR scheduler	Cosine with 10% warmup
Max gradient norm	1.0
Batch size	16 per device
Epochs	10
Max sequence length	64 tokens
Optimizer	AdamW (HuggingFace default)
Precision	bf16
Training loss	Completion-only (prompt tokens masked)

Table 2: LoRA hyperparameters shared across all adapter types. Target modules vary by adapter type: QK = {`q_proj`, `k_proj`}; VO = {`v_proj`}; MLP = {`gate_proj`, `up_proj`, `down_proj`}; Full = all of the above.

A.4 LoRA ADAPTER MATHEMATICS

Each LoRA adapter adds a low-rank perturbation $\Delta W = BA$ ($B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times d}$) to the target weight matrix. The four adapter types perturb distinct parts of the transformer update:

QK (routing). Targets `q_proj` and `k_proj`, perturbing the attention routing weights:

$$\mathrm{Attn}(h) = \mathrm{softmax}\Big(\frac{(W_Q + \Delta W_Q)\,h \cdot \big((W_K + \Delta W_K)\,h\big)^\top}{\sqrt{d_k}}\Big). \tag{5}$$

This modifies *which* tokens attend to which, and hence which basin the trajectory is steered toward, without altering the content read out.

VO (readout). Targets `v_proj`, perturbing the content write-back channel:

$$\Delta_{\mathrm{attn}} = W_O \cdot \mathrm{Attn}(h) \cdot (W_V + \Delta W_V)\,h. \tag{6}$$

The rank-r update $\Delta W_V = B_V A_V$ distorts the content projected into the residual stream within a restricted subspace, without changing routing or basin structure.

MLP (state update). Targets `gate_proj`, `up_proj`, and `down_proj`, perturbing the SwiGLU feed-forward block:

$$\mathrm{MLP}(h) = (W_{\mathrm{down}} + \Delta W_{\mathrm{down}})\big[\sigma\big((W_{\mathrm{gate}} + \Delta W_{\mathrm{gate}})\,h\big) \odot (W_{\mathrm{up}} + \Delta W_{\mathrm{up}})\,h\big], \tag{7}$$

where σ is SiLU and $\odot$ is element-wise multiplication. This modifies the iterated map $h \mapsto h + \mathrm{MLP}(h)$ that shapes basins (Section 3.2).

Full. Targets all of the above modules simultaneously, perturbing routing, readout, and basin structure jointly.

A.5 Evaluation protocol

All evaluations use greedy decoding (`max_new_tokens` $= 8$). We report two accuracy metrics:

- **Exact match:** The generated string exactly equals the target code.
- **Contains:** The target code appears as a substring of the generated response (more lenient, accounts for minor formatting differences).

For knowledge conflict evaluation, we additionally classify each response as:

- **Context wins:** The response contains the context-provided code c_{ctx}.
- **Weights win:** The response contains the parametrically memorized code c_{stored}.
- **Neither:** The response contains neither code (garbage output or unrelated text).

Evaluation set sizes. Table 3 lists the number of queries behind each analysis.

Analysis	Adapters	Situations	Queries
Recall and reading accuracy (Fig. 4a)	brittle	recall / reading / conflict / missing	1,600 / 200 / 200 / 200
Per-digit and mechanistic (Figs. 4b, 5; App. D.2)	brittle	each situation	30
Robust behavioral sets (Fig. 4a)	robust	five held-out sets	50 each
Robust conflict per digit (Fig. 4c)	robust MLP	conflict	30
Geometry and probes (Secs. 3.2, 5; Fig. 6a,b)	robust MLP	recall / novel phrasing / conflict / missing / reading	200 / 50 / 50 / 100 / 50
Scaling relation (Sec. 5.3; Fig. 6c)	12 pretrained models	TriviaQA, NQ-Open	1,500 each
Hallucination rate (App. L)	17 pretrained models	TriviaQA	2,000

Table 3: Number of queries behind each analysis.

B Capacity Scaling Experiments

To understand how memorization efficiency varies with adapter architecture and rank, we conduct systematic scaling experiments. All adapters achieve near-perfect final training loss (≈ 0.0001) across all tested N; the meaningful difference is *learning efficiency*, that is, how many gradient steps are required to memorize N entities.

B.1 Module ablation: scaling entities at rank 8

We train all four adapter types (QK, VO, MLP, Full) at rank $r = 8$ across seven entity counts: $N \in \{25, 50, 100, 200, 400, 800, 1{,}600\}$, each with 100 repetitions per entity. Table 4 reports steps to convergence (first step where training loss < 0.05) and Figure 7a visualizes the scaling curves.

	Number of entities N						
Adapter	25	50	100	200	400	800	1,600
QK	370	690	1,360	3,420	7,090	14,830	30,620
VO	220	480	790	1,890	3,920	8,370	17,240
MLP	170	340	600	1,170	2,200	3,720	8,260
Full	170	300	630	1,100	1,990	3,590	7,880

Table 4: Gradient steps to convergence (first step with training loss < 0.05) by adapter type and entity count ($r = 8$, 10 epochs). All adapters reach near-zero final loss; the table captures learning efficiency. MLP and Full require roughly 3–4$\times$ fewer steps than QK, consistent with MLP being the main substrate for storing associations.

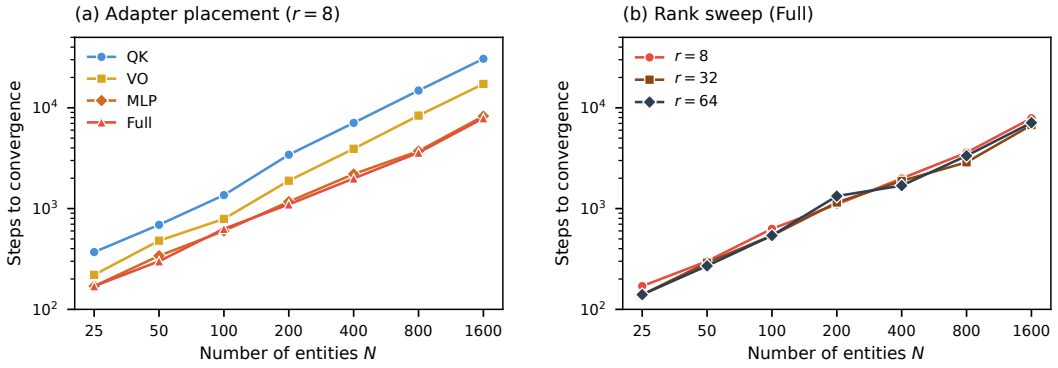

Figure 7: **Learning efficiency scaling curves.** Gradient steps to first reach training loss < 0.05 (log–log scale). All adapters achieve near-zero final loss for all N; the y-axis captures how efficiently each adapter memorizes. *(a)* Adapter placement at $r = 8$. MLP and Full require $\sim$3–4$\times$ fewer steps than QK, consistent with MLP being the main substrate for storing associations. Steps scale approximately linearly with N for all adapters. *(b)* Rank sweep for the Full adapter. Rank has little effect on learning efficiency, with at most $\sim$15% fewer steps at higher rank.

C Format Sensitivity Under Brittle Storage

To quantify the format gating observed under brittle (single-template) storage, we evaluate all four $r = 8$, $N = 1{,}600$ adapters on five prompt formats using 30 held-out entities. Table 5 and Figure 8 report recall accuracy.

Prompt format	QK	VO	MLP	Full
Original training format	100%	100%	100%	100%
Context prefix	100%	53%	63%	20%
Word-order variation	0%	0%	0%	0%
Completion style	0%	0%	0%	0%
Bare entity	0%	0%	0%	0%

Table 5: Recall accuracy across prompt formats under brittle training. All adapters achieve 100% on the exact training template and 0% on any rephrasing of the query. Only QK fully tolerates a context prefix (100%, versus 53%, 63%, and 20% for VO, MLP, and Full), suggesting that addressing is more robust to prompt structure than the content pipeline.

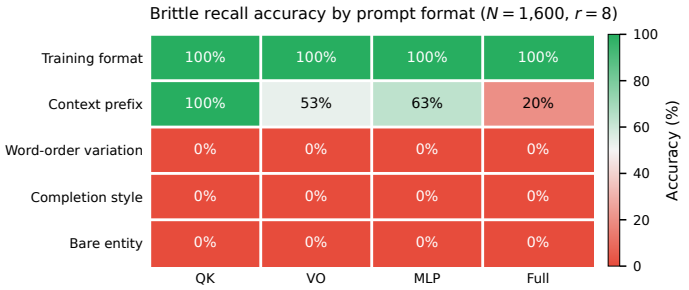

Figure 8: **Format sensitivity under brittle storage.** Recall accuracy across five prompt formats and four adapter placements. Any rephrasing of the query reduces accuracy to 0%. A context prefix is tolerated fully by QK and partially by VO, MLP, and Full.

D CONFLICT ARBITRATION DETAILS

D.1 INSTRUCTION SENSITIVITY

We test whether explicit instructions can influence conflict resolution under brittle storage. Table 6 shows how often the context wins across three instruction conditions.

Adapter	Default	"Use notes"	"Use own knowledge"
QK	100%	93%	100%
VO	93%	97%	97%
MLP	50%	40%	50%
Full	33%	40%	43%

Table 6: Rate at which the context wins under conflict with varying instructions ($N = 30$ per condition, brittle storage). Instructions have minimal effect, and Weights win 0% across all 360 trials. For MLP and Full, the "neither" category (garbage output) accounts for the remainder.

D.2 AGREEMENT BETWEEN WEIGHTS AND CONTEXT

When the context and the weights hold the same code, we compare reading accuracy in two conditions: context only (unseen entities, $N = 200$) and weights and context agree (seen entities where the context-provided and parametrically memorized codes match, $N = 30$). Figure 9a shows aggregate accuracy: QK remains near-perfect under both conditions ($+0.5\%$); MLP shows a substantial rescue ($+24.3\%$, from 39% to 63%); VO degrades sharply (-38.7%, from 92% to 53%); Full degrades modestly (-10%). Figure 9b reveals the per-digit structure: MLP's rescue is concentrated at digit 5, consistent with the stored basin capturing the trajectory late, once enough digits have been generated, while VO's degradation begins at digit 2 and worsens under agreement, consistent with a shared write-back bottleneck.

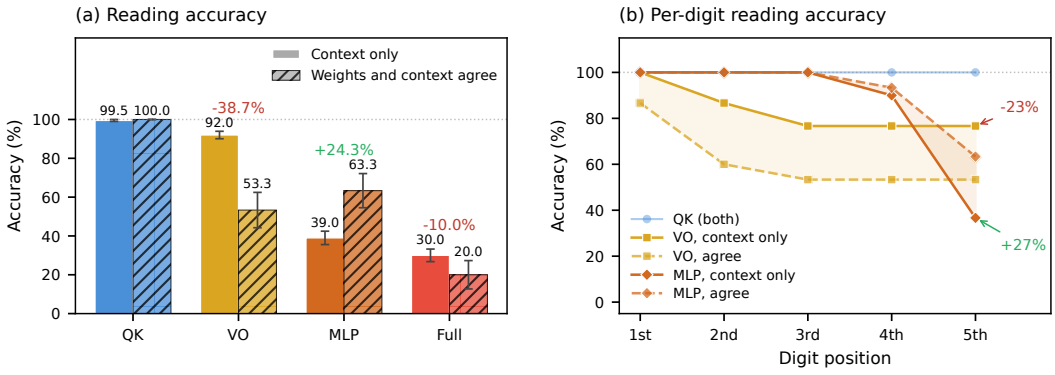

Figure 9: **Agreement between weights and context.** *(a)* Reading accuracy when only the context holds the code (solid) and when the weights and the context hold the same code (hatched). MLP shows the largest rescue ($+24.3\%$), while VO shows the largest degradation (-38.7%). *(b)* Per-digit accuracy for QK (reference), VO, and MLP in the same two conditions (solid and dashed). Shaded regions highlight the difference. MLP rescue is concentrated at digit 5 (prefix-triggered stored basin activation); VO degrades from digit 2 onward (shared write-back bottleneck).

D.3 SAMPLE GENERATION OUTPUTS

Tables 7 and 8 show representative generation outputs that illustrate the qualitative failure signatures of each adapter type. For comparison, without an adapter the base model produces a five-digit code for only 55 of the 1,600 entities it never learned, and otherwise declines, typically with a response such as "I'm sorry, but I don't have any information about…"

Adapter	Entity	Target	Generated
never learned (hallucination): adapter on, entity not in training set			
QK	E002000	62950	`49232`
QK	E002001	59906	`13779`
VO	E002000	62950	`65486`
VO	E002001	59906	`18658`
MLP	E002000	62950	`65325`
MLP	E002003	88569	`95991`
MLP	E002004	33435	`95991`
Full	E002000	62950	`85647`
Full	E002001	59906	`17175`

Table 7: Sample hallucination outputs (never learned). All adapters generate confident 5-digit codes for entities never seen during training. Each digit is independently wrong, indicating hallucination rather than partial retrieval. MLP generates the same code (`95991`) for two different entities, consistent with the output defaulting to a common trained code.

Adapter	Entity	Context code	Generated	Notes
reading (adapter on, code in context, unseen entities)				
MLP	E000000	93810	`93819`	Digit 5 wrong
MLP	E000001	24592	`24599`	Digit 5 wrong
MLP	E000005	39256	`39257`	Digit 5 wrong
Full	E003000	16863	`1686` Fourth Street's...	Text drift at digit 5
Full	E003001	45084	`4508` Fourth and nearly...	Text drift at digit 5
Full	E003004	44937	`4493` Fourth Ward,...	Text drift at digit 5
Knowledge conflict (context and weights provide different codes)				
QK	E000836	98255	`98255`	Context wins (100%)
VO	E000088	34616	`31679`	Neither wins
MLP	E000836	98255	`98259`	context prefix, digit 5 wrong
MLP	E000088	34616	`34619`	context prefix, digit 5 wrong
Full	E000836	98255	`9825` Fourth Ward is...	context prefix, text drift
Full	E000088	34616	`3461` Fourth Ward, New...	context prefix, text drift

Table 8: Sample reading and conflict outputs illustrating adapter-specific failure signatures. **MLP** correctly reads the first 4 digits from context but systematically replaces digit 5 (often with `9`), consistent with the stored basin capturing the trajectory late. **Full** produces 4 correct digits followed by the token "Fourth" (the base model interprets the 4-digit prefix as a street address), then coherent but non-numeric text. **QK** faithfully copies the code in the context. **VO** occasionally produces codes matching neither source. Underlined digits indicate errors. Stored codes are omitted for space; the stored code never wins in any conflict trial.

D.4 Layer-wise conflict trajectory heatmaps

Figure 10 visualises the signed distance $\Delta = \|h_{\text{conflict}} - h_{\text{stored}}\| - \|h_{\text{conflict}} - h_{\text{ctx}}\|$ across all 36 transformer layers and 5 digit-generation positions for both brittle and robust storage MLP adapters under knowledge conflict. The context reference h_{ctx} is obtained by running the same conflict prompt through the base model with the adapter disabled, providing a trajectory that follows reading without any parametric memory installation. Each cell averages over $N = 30$ conflict trials.

E Basin Properties and Controls

Section 3.2 defines a fact's basin as the region around its memorized state m_i into which the hidden state converges. We use the term for three measurable properties.

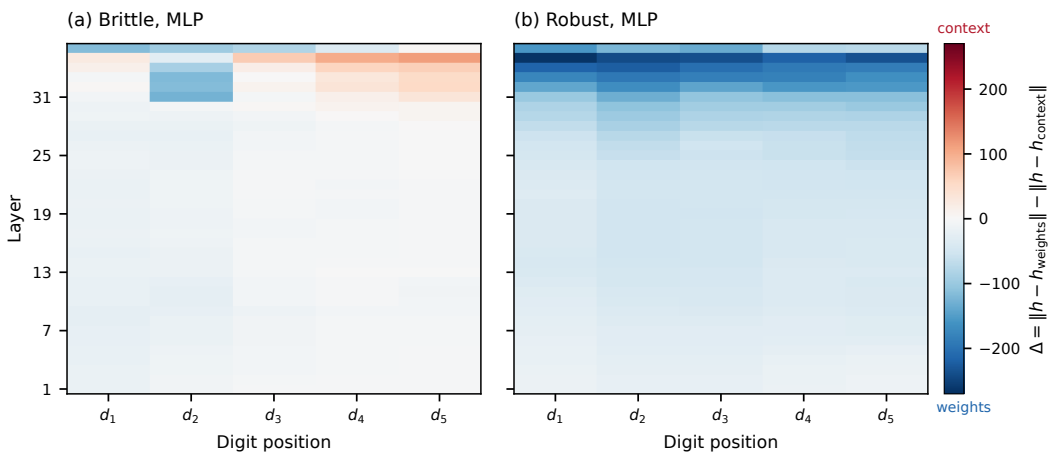

Figure 10: **Signed distance heatmaps under knowledge conflict.** $\Delta = \|h_{\text{conflict}} - h_{\text{stored}}\| - \|h_{\text{conflict}} - h_{\text{ctx}}\|$, where blue means closer to the stored fact and red closer to the context. Both panels share the same colour scale, and values quoted below are means over layers 25–34. *(a)* Brittle storage, MLP adapter. At digit 1, the state lies on the stored side ($\Delta \approx -5$, and -117 at the final layer), even though brittle storage ends up following the context, so the first digit alone does not reveal how the conflict resolves. The state moves toward the context by d_4–d_5 ($\Delta \approx +28$), while the dip at d_2 ($\Delta \approx -56$) corresponds to the stored basin pulling back after the first context digit is generated. *(b)* Robust storage, MLP adapter. The state stays on the stored side at every digit position ($\Delta \approx -120$ to -140), consistent with varied-template training installing basins deep enough to resist the context. Together these panels support the main-text claim that conflict arbitration is a *trajectory* property resolved over multiple generation steps, not a static point property diagnosable from a single first-digit geometric snapshot.

Contraction. Through layers 10–25, about three-quarters of the feed-forward block's symmetric eigenvalues are negative (mean ≈ -0.37), so the MLP update pulls nearby trajectories together. Fine-tuning sustains this contraction into the late layers, where the base model's feed-forward block turns expanding.

Convergence. Under varied-template training, hidden states from a fact's many phrasings end up within about 26 units of its memorized state, while different memorized states sit roughly 130 units apart. This is why the margin separates recall (about 22) from never-learned queries (about 133) so cleanly.

Stability. Under noise added to the input embeddings, the varied-template store keeps producing its answer over a noise range about twice as wide as the single-template store (Appendix F).

Finite-time sense. A basin in this sense is a finite-time basin of the dynamics in Eq. 1, not the basin of a global energy function. The composed map has no such function. Each layer applies a different map, and the composed step is statistically unstructured ($\alpha \approx 0.71$), so there are no fixed points and no landscape that the trajectory descends, only local regions it is pulled into.

Controls. The margin could in principle reflect the identity of the entity string rather than stored knowledge, since the identifiers are lexically distinct. To test this, we compute the memorized states with the base model instead, using the same entities, templates, and computation. Recalled and never-learned queries then sit at 126.2 and 126.4 units, indistinguishable (AUROC ≈ 0.5), so the separation comes from installing the knowledge, not from the identifiers. In a matched comparison of single-template and varied-template adapters, identical except for training phrasing and both recalling 100% of their own phrasing, the single-template margin stays at chance at all 36 layers, while the varied-template margin grows to a gap of $+110$. A single phrasing binds a fact to its surface form without forming a basin, while varied phrasing forces a representation that survives rephrasing.

F PERTURBATION STABILITY OF THE MEMORIZED STORE

To quantify the width of a basin, we measure recall error rate and prediction entropy under Gaussian embedding perturbations. For each of 30 test entities, we add isotropic noise $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ to the input embeddings at every digit position during autoregressive generation, where $\sigma = \alpha\|\mathbf{e}\|$ scales with the mean embedding norm ($\|\mathbf{e}\| \approx 1.02$). We sweep $\alpha \in \{0, 0.005, 0.01, 0.02, 0.05, 0.1\}$ (symmetric positive and negative) with 10 random noise samples per magnitude, yielding 300 trials per point.

Figure 11 shows that the robust MLP adapter maintains a wider stability basin than the brittle adapter:

- **Robust MLP** tolerates $\alpha \leq 0.5\%$ of embedding norm with 0% error and near-zero entropy. Error rises to $\sim$6% at $\alpha = 1\%$ and saturates at $\sim$100% by $\alpha = 2\%$.
- **Brittle MLP** begins failing at $\alpha = 0.5\%$ ($\sim$4% error) and collapses to $\sim$91% error at $\alpha = 1\%$.

The robust adapter's stability window is roughly $2\times$ wider than the brittle adapter's, even though its raw logit gap is slightly smaller ($\sim$17 vs. $\sim$21), suggesting a shallower but broader basin.

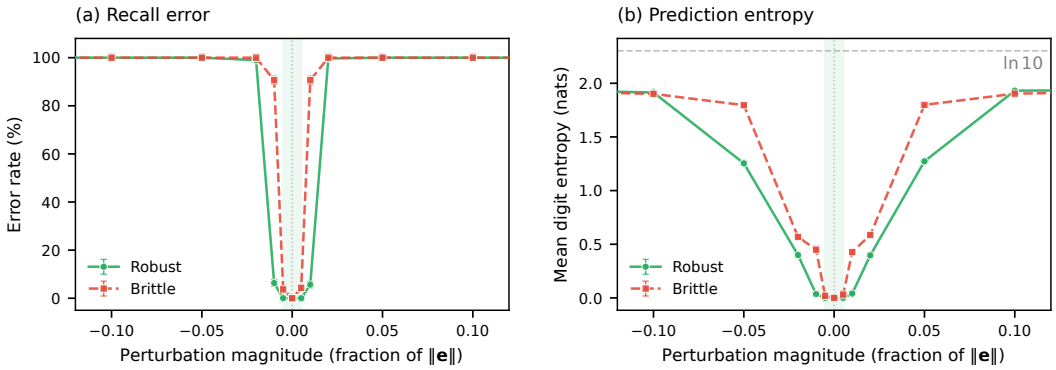

Figure 11: **Perturbation stability of the robust vs. brittle basin.** Gaussian noise ($\sigma = \alpha\|\mathbf{e}\|$) added to input embeddings during autoregressive digit generation. 30 entities $\times$ 10 trials per magnitude point; error bars show SEM. *(a)* Recall error rate. At $\alpha = 1\%$, robust storage (green) errs on 6% of trials and brittle storage (red) on 91%, and both reach 100% by $\alpha = 2\%$. *(b)* Mean digit entropy (nats). Entropy rises with the error, but more gradually, and approaches the uniform value only at $\alpha \geq 10\%$, consistent with perturbation first moving the state out of the basin and only later flattening the output distribution. The dashed line marks $\ln 10$ (uniform over 10 digits).

G METACOGNITION GEOMETRY: EXTENDED RESULTS

G.1 LAYER-WISE ENTROPY TRAJECTORIES

To understand *why* entropy-based detection fails on a subset of hallucinations, we examine how digit entropy evolves across all 36 transformer layers. Figure 12 plots mean digit entropy $\pm$ SEM for three conditions (recall, hallucination on never-learned entities, and reading) for each of the four adapter types.

Several patterns are consistent across adapters:

- **Recall** entropy collapses sharply in layers 25–35, reaching near zero at the output layer. This is consistent with the trajectory settling into the memorized basin.
- **Hallucination** (never learned) entropy also decreases substantially across layers and converges to a low final value for a subset of cases. The hallucinations decompose into two geometric regimes. A minority ($9/100$) exhibit *progressive basin capture*: entity-name token similarity (e.g., querying E002050 initiates routing toward E001050's basin) gives a nearest-basin distance of $\approx$120 at the first-digit position, considerably smaller than the majority ($91/100$, dist $\approx$134) but still far from the recall regime (dist $\approx$22). This distance

gap is expected, since the first-digit measurement precedes autoregressive generation, so the code prefix that would commit the model to the basin has not yet been produced. As generation proceeds, the growing prefix would progressively pull the hidden state closer to the captured basin; the first-digit position captures routing *initiation*, not routing *completion*. The majority $(91/100)$ shows no geometric basin guidance regardless of generation step: the adapter's output-projection biases sharpen the digit distribution without spatial basin routing.

- **Reading** entropy follows a different path from recall, dipping around layer 23 and then collapsing to near zero after layer 30, several layers before recall entropy reaches its final value. This is consistent with the answer being copied from the context rather than retrieved from a basin.

Notably, for the confident-hallucination subset (lowest entropy quartile), the entropy trajectory in final layers (30–35) converges to values similar to recall, explaining why entropy-only classifiers fail on $\sim$13% of cases. Geometric margin does not suffer from this failure: the first-digit hidden state lies far from all stored basins (dist $\approx$105–135) even for confident hallucinations, cleanly separating them from correct recall (dist $\approx$22). The gap signal reinforces this: progressive-capture hallucinations have gap $\approx$17 (one basin is ahead) vs. recall gap $\approx$111 (unambiguous basin winner) vs. unguided hallucinations gap $\approx$9 (equidistant from several basins). Margin and entropy measure different quantities, *proximity to a memorized basin* and *sharpness of the output distribution*, and both types of hallucination produce low entropy through distinct mechanisms (partial basin initiation vs. output-projection bias) while remaining geometrically distinguishable from correct recall at the first-digit measurement point.

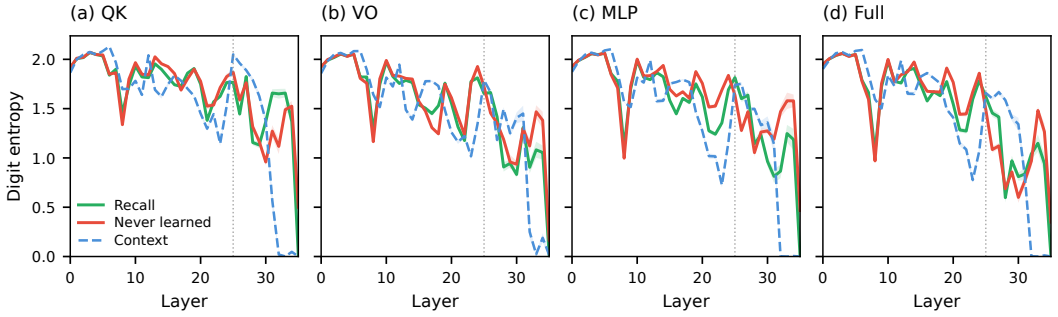

Figure 12: **Layer-wise digit entropy trajectories.** Three conditions across all four adapter types. Mean $\pm$ SEM over 30 entities per condition. The dotted vertical line marks layer 25, where recall entropy (green) begins its sharpest collapse. Entropy on never-learned entities (red) converges to comparably low values in the final layers for a subset of cases, explaining why output-level entropy fails on $\sim$13% of hallucinations. Geometric analysis shows these confident hallucinations are not in any stored basin (nearest-basin distance $\approx$105–135, vs. $\approx$22 for recall), so the margin identifies them even when entropy does not. Entropy when reading from the context (blue, dashed) follows a different path and collapses to near zero only after layer 30.

H Attention-Mediated Head Selection and VO Symmetry

The mid-layer VO symmetry peak reported in Section 3 is not a static property of the weight matrices W_O and W_V: computing $\varphi(W_O W_V)$ from weights alone yields $\varphi \approx 0.02$ uniformly across layers. Instead, the symmetry emerges dynamically through *attention-mediated head selection*.

Mechanism. The VO Jacobian at the last token position decomposes as

$$J_{\text{VO}} = \sum_{h=1}^{H} a_{t,t}^{h} \, W_O^h \, W_V^h,$$

where $a_{t,t}^h$ is the self-attention weight of head h at position t. In mid-layers, the attention sink phenomenon Xiao et al. (2024) concentrates most attention mass on the first token, driving $a_{t,t}^h \approx 0$

for the majority of heads. The surviving heads, those with high self-attention weight, happen to have highly symmetric per-head weight products $\varphi(W_O^h W_V^h) \gg 0$.

Figure 13(a) shows this relationship at layer 15: the correlation between per-head self-attention weight and per-head φ is $r = 0.88$. The attention-weighted sum achieves $\varphi = 0.64$, a $26\times$ boost over the uniform-weighted sum ($\varphi = 0.02$; Fig. 13b). This boost peaks at mid-layers where attention sinks are strongest, explaining the layer profile of VO symmetry.

Cross-model replication. The mid-layer VO symmetry peak is not specific to Qwen2.5-3B. Figure 13(c) shows peak φ across five model families (Qwen, Llama, OLMo, Mistral, StableLM) spanning 0.5B–7B parameters. All families exhibit a mid-layer φ peak, though the magnitude decreases with model size ($r = -0.998$ within the Qwen family). Interestingly, the area under the φ-versus-layer curve remains approximately constant across these models, suggesting that larger models may distribute this symmetric VO structure across more layers.

Interpretation. Recent work (Sun et al., 2026) finds that attention sinks bias individual heads toward short-range dependencies, arise largely as a byproduct of short-context training, and are strongly suppressed when attention is gated by a function of the current representation. Whatever their origin, sinks have a consequence for VO geometry that matters here: by zeroing out the self-attention weights of dormant heads ($a_{t,t}^h \approx 0$), the sink means the effective VO Jacobian is dominated by the heads that retain high self-attention weight, which also exhibit more symmetric $W_O^h W_V^h$ products. The correlation between $a_{t,t}^h$ and per-head φ ($r = 0.88$) suggests that heads that avoid becoming sinks tend to have more symmetric $W_O^h W_V^h$ products. One possible interpretation is that non-sink heads apply a symmetric $W_O^h W_V^h$ via self-attention, which acts as the gradient of a quadratic potential $\nabla(\frac{1}{2}\tilde{h}^\top M^{(h)}\tilde{h})$ and could contract the hidden state toward stored patterns. Sink heads, whose attention mass falls on the first token (a near-constant vector after normalization Sun et al. (2026)), contribute only a fixed bias that does not participate in retrieval dynamics. Under this interpretation, MLP shapes basins, non-sink VO heads perform the within-basin readout, and sink heads are effectively gated off, which would be consistent with the steer/shape-basins/read-out roles of Section 3.

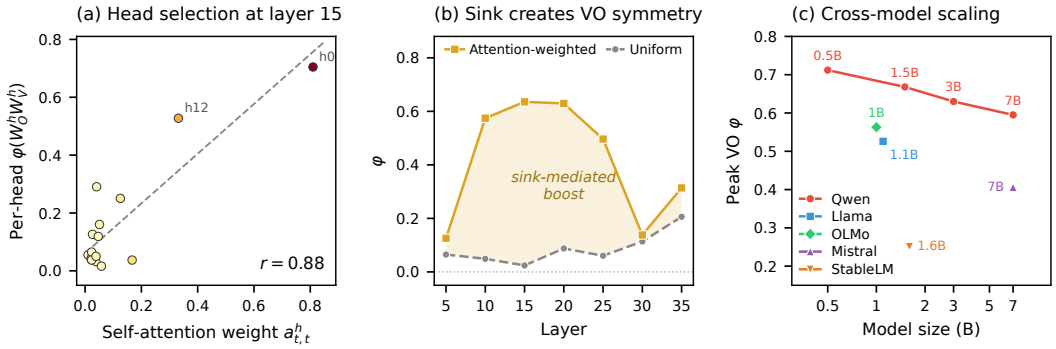

Figure 13: **Attention-mediated head selection creates VO symmetry.** *(a)* Per-head scatter at layer 15: heads with high self-attention weight $a_{t,t}^h$ have highly symmetric $W_O^h W_V^h$ ($r = 0.88$). Head 0 dominates with $a_{t,t} = 0.81$, $\varphi = 0.71$. *(b)* Attention-weighted φ (gold) vs. uniform-weighted φ (grey dashed) across layers. The shaded region shows the sink-mediated boost, which peaks at mid-layers where attention sinks are strongest. *(c)* Peak VO φ vs. model size across five families. All models exhibit mid-layer symmetry; peak magnitude decreases with size while integrated φ remains approximately constant across these models.

I End-to-End Metacognitive Head: A Negative Result

A natural question is whether metacognition can be *trained in* rather than architecturally provided: add a small MLP head that predicts $P(\text{correct})$ from the hidden state and train it jointly with the backbone, so gradients reshape representations to support self-monitoring. We test this on our toy setup.

Setup. We add a 3-layer MLP (2048→256→256→1) to the model's last-layer hidden state at the first-digit position. Training uses 300 queries (200 seen entities + 100 unseen) with binary correctness labels obtained from a single pre-pass. We compare three conditions: (A) post-hoc linear probe on frozen hidden states, (B) post-hoc MLP on frozen hidden states, and (C) end-to-end MLP with LoRA gradient flow. For condition C, we freeze LoRA for the first epoch (head-only warmup), then jointly train for 4 additional epochs with a combined loss $\mathcal{L} = \lambda \cdot \mathcal{L}_{\text{BCE}} + (1 - \lambda) \cdot \mathcal{L}_{\text{LM}}$ ($\lambda = 0.3$, LoRA LR $= 10^{-6}$) to prevent catastrophic forgetting.

Method	AUROC	Correct preserved (%)
Entropy (output-based)	0.967	0.5
Linear probe (post-hoc)	0.964	—
MLP head (post-hoc)	0.953	18.0
MLP head (end-to-end)	0.913	0.5
Oracle margin	0.994	100.0

Table 9: Metacognitive head comparison. "Correct preserved" is the fraction of correct recalls retained when thresholding to catch 100% of hallucinations (recall vs. hallucination, N=300).

Results. End-to-end training *degrades* performance relative to post-hoc probes (AUROC 0.913 vs. 0.953). Despite the LM loss anchor, backbone gradients from the metacognitive head distort the basin geometry: the correlation between pre- and post-training margins drops to $r = 0.30$, and margin AUROC falls from 0.994 to 0.728. The head learns to predict correctness on the training distribution (98% accuracy) but on shifting representations, so by the time it converges, the geometric structure it depends on has changed.

Interpretation. This negative result is informative: even with LM loss regularization and a conservative LoRA learning rate, the basin geometry that makes margin work, which is a *consequence* of the knowledge stored in the weights, is disrupted by gradients from the metacognitive head. The head achieves near-perfect training accuracy (98%) but on shifting representations; by the time it converges, the geometric structure it depends on has changed. While more sophisticated training schedules (e.g., alternating updates, gradient projection) might mitigate this interference, this experiment reveals a tension in this training setup: gradients that reshape representations for self-monitoring also reshape the knowledge itself. Robust metacognition may therefore benefit from access to knowledge-base geometry that does not depend on end-to-end training of a correctness predictor alone, and the geometric distillation below partially recovers the gap.

I.1 Geometric distillation: a partial recovery

The negative result above trained the head on binary correctness labels, a signal with no geometric inductive bias. We hypothesized that explicitly distilling the margin and gap computations into the head would avoid the collapse, because the head would learn to approximate the nearest-neighbor distance function parametrically rather than memorizing a correctness lookup.

Three-phase training. We co-train a fresh LoRA adapter and a dual-output metacognitive head (shared backbone, separate geometric and confidence output layers) in three phases:

1. **Phase 1 (epochs 1–3): Knowledge installation only.** LoRA trains with standard LM loss; the head receives no gradient. This lets basin structure form before the head begins learning, avoiding the degenerate all-incorrect labels that caused collapse in the original e2e attempt.
2. **Phase 2 (epochs 4–7): Geometric distillation.** Basin centers are computed from the current LoRA representations (averaged across 3 canonical templates per entity). The head is trained with MSE to predict normalized margin $\hat{\delta}(x)$ and gap $\widehat{\text{gap}}(x)$ from the hidden state alone. Basin centers are refreshed every 2 epochs to track evolving representations. LoRA continues training with LM loss; the combined loss is $\mathcal{L} = 0.8 \cdot \mathcal{L}_{\text{LM}} + 0.2 \cdot \mathcal{L}_{\text{geo}}$.
3. **Phase 3 (epochs 8–10): Calibration.** LoRA is frozen. The head is fine-tuned with BCE on binary correctness labels (obtained via generation), converting geometric awareness into a usable $P(\text{correct})$ estimate. Because the head already encodes margin-like representations from Phase 2, it does not collapse to a trivial classifier.

Method	AUROC	Correct preserved (%)
Entropy (output-based)	0.623	5.0
MLP head, BCE labels (e2e, §I)	0.913	0.5
Post-hoc probe (logistic regression)	0.982	—
Geo-distilled head: confidence mode	0.932	2.0
Geo-distilled head: predicted margin	0.898	58.5
Oracle margin (NN lookup)	0.993	100.0

Table 10: Geometric distillation results ($N = 450$). "Correct preserved" is the fraction of correct recalls retained when thresholding to catch 100% of hallucinations ($N_{\text{recall}} = 200$, $N_{\text{hall}} = 100$). The geo-distilled head's predicted margin preserves 58.5% of correct recalls, compared with 0.5% for the end-to-end head and 5.0% for entropy.

Results. The geometric distillation substantially improves over the original e2e attempt. The calibrated confidence output achieves AUROC 0.932 (vs. 0.913 for BCE-only e2e), but the main result is the predicted margin output: while its raw AUROC is 0.898, the intervention metric (preserving correct recalls while catching all hallucinations) improves from 0.5% to **58.5%**. The head has learned a margin-like function that, while noisier than the oracle, captures the geometric separation between known and unknown regions of representation space.

Per-condition analysis. The head assigns high confidence to conflict queries (1.000, identical to recall), consistent with it having learned a margin-like signal: conflict entities *are* near a memorized basin, so the head correctly reports proximity. The failure mode under conflict is basin *competition*, not basin *absence*, and a margin-based signal cannot distinguish these, consistent with the main-text analysis (Section 4.1).

Interpretation. The phased geometric distillation avoids the representation-collapse problem of the original e2e attempt by (i) letting basins form before the head trains, and (ii) providing a continuous geometric target (margin, gap) rather than binary labels that degenerate in early training. The head learns an approximation of the NN-distance function parametrically: at inference, it requires only a single forward pass through a small MLP, with no stored basin centers and no nearest-neighbor lookup. This partially closes the scalability gap identified in the main text discussion. However, a substantial gap to the oracle remains (AUROC 0.932 vs. 0.993; intervention 58.5% vs. 100%), and this result motivates examining when the relevant geometric signal becomes available.

I.2 Layer-wise emergence of basin geometry

A natural question for any metacognitive architecture is: at which network depth does the geometric signal become available? If margin is informative at intermediate layers, a mid-layer module could feed this information back into the residual stream for downstream layers to act on, allowing the model to condition its processing on whether it knows the answer. If the signal emerges only at the final layers, such feedback is architecturally infeasible within a single forward pass.

Setup. For each of the 37 layers (embedding + 36 transformer layers), we compute basin centers (1,600 entities $\times$ 3 templates, averaged) and evaluate margin AUROC on the standard 450-query test set. All computations use the co-trained adapter from the geometric distillation experiment.

Results. Three findings are notable:

1. **Basin geometry is a late-layer phenomenon.** Layers 0–18 carry little margin signal (AUROC < 0.80). The earliest layer exceeding 0.95 AUROC on recall vs. hallucination is layer 24, two-thirds of the way through the network.
2. **Peak discrimination occurs at layers 28–30**, not the final layer. Layers 29–30 achieve AUROC 1.000 for recall vs. hallucination with margin separation of 10–11 units.
3. **The signal degrades in the final layers** (33–36), as representations transition from basin-structured hidden states to output-projection space. Layer 36 drops to AUROC 0.725, worse than layer 12. This degradation is specific to the nearest-basin margin: correctness remains linearly decodable from the final representation (Section 5.1), indicating that late layers transform rather than erase the relevant information.

Implications for metacognitive architecture. The layer-wise profile suggests a constraint on mid-layer metacognitive feedback in this architecture. The geometric signal becomes reliably available only at layer ≈ 24, leaving 12 layers (one-third of the network) to act on it. However, the attention that selects between the weights and the context appears to operate mainly in earlier layers, where the signal does not yet exist. The model cannot condition its source-selection on epistemic state because that state has not yet been computed.

This suggests a timing constraint for single-pass architectures, in which the signal that would enable informed arbitration is produced by the same computation it would need to steer. The model must process the input deeply enough to determine whether it "knows" the answer, but by that depth, the processing decisions (attention routing, source selection) have already been made. This offers one possible explanation for why end-to-end training of metacognitive heads recovers only part of the gap: in a single forward pass of this architecture, the signal may arrive too late to influence the computation that produces the output. We present this as a hypothesis; other architectures, auxiliary objectives, or cross-layer pathways could make the signal available earlier.

Multi-pass architectures, in which the model first assesses whether it knows the answer and then re-processes the input conditioned on that assessment, are one way this constraint could be avoided.

J Extended Related Work

This section expands on the related work discussion in Section 6 of the main text.

Knowledge storage and localization in transformers. Geva et al. Geva et al. (2021) showed that MLP layers function as key–value memories, whose keys detect input patterns and whose values promote output distributions, composed across layers via the residual stream. Subsequent work on knowledge editing Meng et al. (2022; 2023) demonstrated that individual factual associations can be localized to specific MLP layers and surgically modified. Our work extends this picture from individual facts to the *collective geometry* of many memorized associations, showing that MLP adaptation creates basins, in the finite-time sense of Section 3.2, whose competition with context governs arbitration and failure modes.

Parameter-efficient fine-tuning. LoRA Hu et al. (2022) and related methods Li & Liang (2021); Houlsby et al. (2019) enable targeted adaptation of pretrained models. While most work evaluates adaptation quality (task accuracy, sample efficiency), we use adapter placement as a *causal probe*: by restricting LoRA to specific component groups (QK, VO, MLP), we isolate the functional contribution of each architectural component to memory storage, revealing qualitatively distinct perturbation signatures.

Hallucination and calibration. LLM hallucination has been studied extensively Ji et al. (2023); Zhang et al. (2025), with proposed mitigations including retrieval augmentation Lewis et al. (2020), sampling-based consistency checks Wang et al. (2023); Kuhn et al. (2023), and post-hoc calibration Guo et al. (2017). Most detection methods score hallucination with a single output-level signal. Our basin account provides a mechanistic decomposition: hallucinations occur when queries land far from all memorized basins yet the LM head defaults to trained output sequences, producing confident outputs indistinguishable from correct recall at the output level. Cross-entity analysis indicates that code overlap with the training set is predominantly an output-layer bias ($\sim 84\%$ of matched codes do not correspond to the nearest basin), not geometric basin capture. This explains why standard calibration methods fail on the most dangerous errors: entropy reflects output-distribution sharpness, not basin proximity.

Mechanistic interpretability. The circuits-based approach to interpretability Elhage et al. (2021); Olsson et al. (2022) has identified attention heads, such as induction heads, involved in specific computations, and feed-forward layers have been characterized as key–value memories (Geva et al., 2021; Meng et al., 2022). Our work complements this by analyzing the *representation-level* consequences of circuit modifications: rather than tracing information flow through individual components, we characterize how adapter-induced changes to circuits reshape the basins in which generation unfolds.

Dynamical systems perspectives on transformers. Several authors have drawn connections between transformer computation and dynamical systems Geshkovski et al. (2025), viewing layers as

time steps of an interacting-particle dynamical system. We apply a related view to the generation trajectory, treated as an input-conditioned iterated map. Our empirical results are consistent with this view without requiring a fixed-point or global-energy structure the composed map does not have (Section 3.2): we observe basins with measurable margins, systematic cross-entity retrieval driven by output-layer bias, and autoregressive trajectories pulled toward or away from a basin over the course of generation.

K A Withdrawn Natural-Language Extension

An earlier version of this appendix extended the margin result to natural-language factual knowledge in the pretrained model (national capitals, languages, continents, and TruthfulQA). We do not report those numbers: the apparent separation was confounded by surface form and by relation base rates rather than by basin geometry, and it did not survive a matched control. The scope of the margin result in this paper is the synthetic task of Section 2, where entity–code association is exact and unconfounded; extending it to open-domain text remains open (Section 7).

L Hallucination Scaling Across Model Families

To test whether representational geometry underlying hallucination generalizes beyond Qwen2.5-3B, we evaluate 17 pretrained instruct models spanning six families (Qwen 2.5: 0.5B–72B; Falcon 3: 1B–10B; SmolLM2: 0.36B–1.7B; OLMo 2: 1B–7B; Mistral: 7B; Phi-4: 14B) on 2,000 TriviaQA Joshi et al. (2017) questions. For each model, we extract hidden states at $\sim$75% network depth, compute hallucination rate, output entropy, and linear probe AUROC (5-fold CV logistic regression on hidden states classifying correct vs. incorrect).

Hallucination rate. Hallucination rate follows a power law across all families: $H \propto N^{-0.27}$ ($r^2 = 0.89$, $p < 0.001$; Fig. 14). Per-family exponents are consistent: Qwen $\alpha = -0.23$, Falcon $\alpha = -0.28$, SmolLM2 $\alpha = -0.18$. Different families have different intercepts, which likely reflect training data and recipe, but converge at larger scales: at 7B parameters, Qwen, Falcon, OLMo, and Mistral all cluster within 30–46% hallucination rate. The similar empirical slopes across these model families suggest that the trend is not specific to a single architecture or training recipe.

Confident hallucination concentrates with scale. Despite lower overall error rates, larger models produce a higher fraction of *confident* wrong answers (output entropy < 0.1): 0.3% of errors at 0.5B vs. 7.8% at 7B. On this confident subset, output entropy carries no signal (AUROC 0.54), while linear probes on hidden states partially recover detection (AUROC 0.75; trained on non-confident examples, tested on the confident subset). The gap (+0.20 AUROC) shows that hidden-state representations contain information useful for distinguishing these errors even when output entropy does not. The remaining gap to perfect discrimination may reflect limitations of the linear probe, the selected layer, or genuinely weaker internal error signals.

L.1 Derivation of the scaling relation $C = \exp(-c/\bar{\Delta})$

We derive Eq. 4 from the softmax entropy function and the statistics of the top-two logit gap. The derivation has three steps.

Step 1: Entropy is a monotone function of the top-2 logit gap. Let $\Delta := \ell_{(1)} - \ell_{(2)}$ be the gap between the top two logits. For any entropy threshold H_0, there exists an empirical cutoff $\Delta^*(H_0, V)$ such that

$$H_{\text{out}}(\ell) < H_0 \iff \Delta > \Delta^*(H_0, V). \tag{8}$$

The 2-class approximation ($H \approx \Delta e^{-\Delta}$) underestimates the required gap by $\sim 1.4\times$ at realistic vocabulary sizes because $\sim$30,000 lower-ranked logits contribute non-negligibly. Empirically, for $V \approx 30{,}000$ and $H_0 = 0.1$, $\Delta^* \approx 5.0$; the 2-class formula predicts 3.57. Table 11 reports this discrepancy across Δ ranges.

Step 2: Confident fraction is a tail probability. Let $\mathcal{D}$ be the distribution of Δ over wrong-answer queries. Then:

$$C := \Pr_{\Delta \sim \mathcal{D}}[\Delta > \Delta^*(H_0, V)]. \tag{9}$$

C is the tail probability of the gap distribution above the vocabulary- and threshold-dependent cutoff Δ^*.

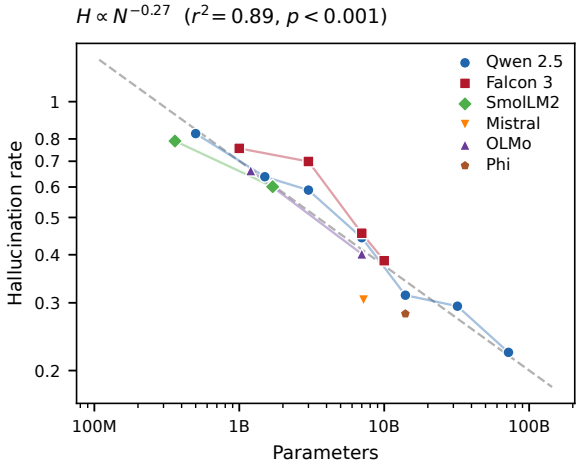

Figure 14: **Hallucination scaling across model families.** Hallucination rate vs. parameter count (log–log) for 17 models across six families. The dashed line is the fit $H \propto N^{-0.27}$ ($r^2 = 0.89$, $p < 0.001$). Lines connect models within a family, which follow similar slopes with different intercepts.

Δ range	Mean H_{out} (empirical)	2-class $\Delta e^{-\Delta}$	Ratio
2.0–2.5	1.39	0.24	5.9
3.0–3.5	0.68	0.13	5.4
4.0–4.5	0.23	0.061	3.8
5.0–5.5	0.088	0.028	3.2
6.0–6.5	0.033	0.012	2.7
7.0–7.5	0.013	0.0051	2.5

Table 11: Mean output entropy on real LLM wrong-answer queries vs. the 2-class approximation $\Delta e^{-\Delta}$. The 2-class formula underestimates entropy by 3–6$\times$ across the relevant range because lower-ranked logits at $V \approx 30{,}000$ contribute substantially.

Step 3: The gap distribution is approximately exponential. We measure the Δ distribution on wrong-answer queries for representative models (Table 12). The std/mean ratio is close to 1 (1.06–1.24), consistent with an exponential distribution.

Model	$\bar{\Delta}$	std/mean	KS-test p vs. exponential
Qwen2.5-0.5B	0.86	1.06	2.4×10^{-3}
Qwen2.5-3B	1.59	1.24	6.1×10^{-4}
Qwen2.5-14B	3.01	1.14	9.2×10^{-2}

Table 12: Logit-gap distribution statistics on wrong-answer queries. std/mean $= 1$ for an exact exponential; empirical values (1.06–1.24) are close. Deviations are the source of the remaining $\sim$17% discrepancy between the predicted constant $c = \Delta^* = 5.0$ and the empirical fit $c \approx 5.87$.

For an exponential distribution with mean $\bar{\Delta}$, $\Pr[\Delta > x] = \exp(-x/\bar{\Delta})$. Substituting:

$$C \approx \exp\left(-\frac{\Delta^*(H_0, V)}{\bar{\Delta}}\right) = \exp\left(-\frac{c}{\bar{\Delta}}\right), \tag{10}$$

recovering Eq. 4 with $c = \Delta^*(H_0, V)$. Using the empirical $\Delta^* = 5.0$ for $H_0 = 0.1$, the predicted slope is -5.0; the empirical through-origin fit gives -5.87 (agreement within 17%, attributable to minor super-exponential tails).

Scaling with model size. We sketch one heuristic account of how $\bar{\Delta}$ could grow with model size, which rests on two assumptions. First, under compute-optimal training the number of training tokens

τ grows roughly in proportion to N. Second, the logit gap grows as $\tau^{1/3}$, as the inverse temperature and the logit standard deviation do under softmax and cross-entropy training (Liu et al., 2026). That work ties this growth to training time rather than model size, and the first assumption need not hold within a family trained on a common token budget, so this sketch is only heuristic. Our main-text claim that larger models have wider gaps rests on the measurements below. Together these give:

$$\bar{\Delta}(N) \propto N^{1/3} \cdot \bar{\Delta}_0. \tag{11}$$

Substituting gives $\log C(N) \approx -c/(\bar{\Delta}_0 \cdot N^{1/3})$. The direct evidence is empirical: within the Qwen 2.5 family, fits give $\bar{\Delta} \propto N^{1/3}$ with $r^2 \geq 0.94$.

L.2 Per-model empirical verification ($r^2 = 0.88$, parameter-free)

Table 13 reports per-model accuracy and confident fraction $C_{0.1}$ on TriviaQA (Joshi et al., 2017) and NQ-Open (Kwiatkowski et al., 2019; Lee et al., 2019). Table 14 verifies the scaling relation: for each (model, benchmark) pair, the predicted $\log C = -\Delta^*_{\text{model}}/\bar{\Delta}$ uses no free parameters, yet matches the empirical $\log C$ with mean ratio 0.96 ± 0.11 across all 21 points ($r^2 = 0.88$).

		TriviaQA			**NQ-Open**		
Model	N (B)	Acc (%)	Wrong	$C_{0.1}$ (%)	Acc (%)	Wrong	$C_{0.1}$ (%)
SmolLM2-360M	0.4	21.8	1173	0.1	9.8	1353	0.0
Qwen2.5-0.5B	0.5	17.7	1234	0.3	9.0	1365	0.1
Falcon3-1B	1.0	25.1	1124	0.2	12.9	1306	0.2
Qwen2.5-1.5B	1.5	36.9	947	2.0	17.3	1241	1.0
SmolLM2-1.7B	1.7	40.6	891	0.1	18.1	1228	0.0
Falcon3-3B	3.0	30.6	1041	1.4	15.3	1271	3.1
Qwen2.5-3B	3.0	40.2	897	5.5	19.5	1208	11.8
Falcon3-7B	7.0	54.5	683	2.5	24.1	1138	1.1
Qwen2.5-7B	7.0	55.0	675	7.1	26.9	1096	13.3
Mistral-7B-v0.3	7.2	69.9	451	20.2	31.2	1032	11.9
Falcon3-10B	10.0	61.6	576	6.4	26.1	1109	9.5
Qwen2.5-14B	14.0	68.0	480	21.2	34.1	989	27.1

Table 13: Per-model results on TriviaQA and NQ-Open (1,500 questions each). $C_{0.1}$: fraction of wrong answers with output entropy < 0.1. SmolLM2 models show near-zero C at all scales because their smaller LM head produces weaker logit sharpening.

L.3 Controlled teacher-student ablation: margin separation vs. confident fraction

To isolate the effect of width in a controlled setting (fixing architecture and training data), we run a teacher-student toy model: a 2-layer MLP teacher with inverse temperature β^* trains a student MLP of varying width m to memorize 500 entity-code associations, then evaluates on 200 held-out entities.

The divergence between δ_u/δ_s and H_{unseen} is the central observation. Wider models produce a *better* geometric signal and a *worse* output signal at the same time. This controlled experiment shows that width alone is sufficient to produce the divergence between geometric and output signals in this toy setting.

Model	Benchmark	$\bar{\Delta}$	Δ^*_{model}	$C_{0.1}$	predicted $\log C$	ratio
SmolLM2-360M	TriviaQA	1.25	6.81	0.001	-5.46	0.77
Qwen2.5-0.5B	TriviaQA	0.86	5.50	0.003	-6.40	1.12
Qwen2.5-0.5B	NQ-Open	0.95	6.00	0.002	-6.29	0.96
Falcon3-1B	TriviaQA	1.06	5.62	0.002	-5.31	0.84
Falcon3-1B	NQ-Open	1.26	5.38	0.002	-4.27	0.66
Qwen2.5-1.5B	TriviaQA	1.15	5.00	0.020	-4.34	1.11
Qwen2.5-1.5B	NQ-Open	1.21	5.25	0.010	-4.32	0.93
Falcon3-3B	TriviaQA	1.39	5.69	0.013	-4.11	0.94
Falcon3-3B	NQ-Open	1.69	5.44	0.030	-3.22	0.92
Qwen2.5-3B	TriviaQA	1.59	5.25	0.049	-3.30	1.09
Qwen2.5-3B	NQ-Open	2.22	4.75	0.117	-2.14	0.99
Falcon3-7B	TriviaQA	1.44	5.38	0.024	-3.72	1.00
Falcon3-7B	NQ-Open	1.35	5.12	0.012	-3.79	0.86
Qwen2.5-7B	TriviaQA	1.96	5.50	0.064	-2.80	1.01
Qwen2.5-7B	NQ-Open	2.38	5.00	0.137	-2.11	1.06
Mistral-7B-v0.3	TriviaQA	2.61	4.88	0.184	-1.87	1.11
Mistral-7B-v0.3	NQ-Open	2.36	4.88	0.121	-2.07	0.98
Falcon3-10B	TriviaQA	1.82	5.44	0.048	-2.99	0.98
Falcon3-10B	NQ-Open	2.01	4.75	0.095	-2.36	1.00
Qwen2.5-14B	TriviaQA	3.01	4.75	0.198	-1.58	0.97
Qwen2.5-14B	NQ-Open	4.02	4.62	0.278	-1.15	0.90

Table 14: Per-(model, benchmark) verification of Eq. 4. The predicted $\log C = -\Delta^*_{\text{model}}/\bar{\Delta}$ uses *no free parameters*: Δ^*_{model} is the model's own empirical entropy-crossing threshold, $\bar{\Delta}$ is its mean logit gap on wrong answers. The ratio of predicted to actual $\log C$ is close to 1 across all 21 points (mean 0.96, std 0.11), collapsing four model families and two benchmarks onto a single line.

m	Params	Acc	H_{seen}	H_{unseen}	δ_u/δ_s	$C_{0.1}$
16	5,008	0.90	0.35	0.29	$1.2\times$	0.52
32	15,136	0.78	0.49	0.14	$5.6\times$	0.67
64	50,752	0.66	0.57	0.06	$43.8\times$	0.88
128	183,424	0.62	0.41	0.02	$103.9\times$	0.96
256	694,528	0.55	0.26	0.004	$152.5\times$	0.99

Table 15: Toy model width sweep (fixed 100K training steps). δ_u/δ_s: ratio of unseen-entity to seen-entity margin, measuring geometric separation. $C_{0.1}$: confident fraction at entropy threshold 0.1. As width grows $16\times$ (from $m = 16$ to $m = 256$), margin separation improves from $1.2\times$ to $153\times$ while C rises from 52% to 99%, so the hidden-state geometry becomes more informative as the output entropy becomes less so.