MEMORY CONTROL SHAPES ADVERSARIAL SENSITIVITY AND SPECIALIZATION IN RECURRENT LANGUAGE MODELS*

Anonymous authors
Paper under double-blind review

ABSTRACT

We show that recurrent memory control shapes not only how much adversarial damage is produced, but also which perturbations are effective. Using Gated DeltaNet-2 (GDN2), which explicitly decouples memory erase from write, together with corresponding controls over new information input and old state decay in Mamba-1 and Hymba, we study these effects under adaptive white box attacks in continuous embedding space. First, write control has a substantially larger effect on adversarial damage than erase across all three architectures. This suggests that inputs which naturally induce stronger write activity may allow adversarial perturbations to have greater downstream influence. In GDN2, this ordering is opposite to the clean performance ablation trend, where erase plays the larger role. Second, changing memory dynamics can weaken a fixed attack without reducing vulnerability to an adapted attack. In our GDN2 post attack erase experiment, about 95% of the apparent recovery comes from the original attack transferring poorly to the modified memory condition, rather than from lower damage under an adapted attack. Finally, we show how this loss of proportional transfer develops during optimization. As optimization proceeds, attacks become stronger on their source memory condition but retain a smaller fraction of source damage after transfer to other conditions. Along the same trajectories, gradients under different memory conditions become less aligned and the corresponding optimized perturbations become less similar. We refer to this process as *memory condition specialization*. Overall, recurrent memory dynamics shape both adversarial sensitivity and attack specificity, highlighting the need to adapt attacks to the target memory condition when evaluating robustness.

1 INTRODUCTION

Recent sequence models increasingly use recurrent state to compress past information instead of relying only on full attention over the entire context. At the same time, recurrent state creates a new route for adversarial influence. A perturbation introduced over a short input span can enter the internal state and affect predictions long after the perturbation itself has ended. In this work, perturbations are optimized directly in continuous embedding space rather than as discrete tokens. Recent work has begun to expose adversarial vulnerabilities in recurrent and state space models, including error amplification under input dependent SSM dynamics and attacks that overwrite or collapse hidden state memory (Le Mercier et al., 2026; Bonetto, 2026). However, it remains unclear which recurrent memory operations control this vulnerability.

Gated DeltaNet-2 (GDN2) provides a useful opportunity to study this question. GDN2 explicitly separates memory erase from memory write (Hatamizadeh et al., 2026):

$$S_t = \left[I - k_t(b_t \odot k_t)^\top\right] D_t S_{t-1} + k_t(w_t \odot v_t)^\top. \tag{1}$$

Here, S_t is the recurrent state, k_t and v_t are the key and value, and D_t is the decay operator. The erase gate b_t controls how existing memory associations are modified, while the write gate w_t controls

*Code and models are available at `https://anonymous.4open.science/r/AR_recurrent-FEB5/README.md`

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.

how new information enters memory. This separation allows us to ask whether write and erase have comparable effects on adversarial damage, or whether one operation dominates.

We find a strong asymmetry. In GDN2, under matched interventions, adaptive adversarial damage is substantially more sensitive to write strength than to erase strength. We observe the same ordering in Mamba-1 and Hymba after separating new information input from old state decay in their recurrent updates. Interestingly, the original GDN2 ablations show the opposite asymmetry on standard tasks: channel-wise erase contributes more to clean performance than channel-wise write.

The write and erase asymmetry leads to a broader question. Does changing memory dynamics only change how much damage an attack causes, or can it also change which attack is effective? We study this question first by holding an adversarial perturbation fixed while changing the memory condition. In GDN2, increasing erase strength after the attack span allows the model to partially recover from the residual effect of a fixed perturbation introduced during the attack span. However, this recovery largely disappears once the attack is adapted to the modified memory condition. Approximately 95% of the apparent recovery comes from attack mismatch, where the source attack loses effectiveness after transfer to the target memory condition, while adaptive attack damage changes very little.

This raises a further question: is this mismatch already present at the start of optimization, or does it grow as the attack becomes better adapted to its source condition? Across GDN2, Mamba-1, and Hymba, we find that deeper optimization makes attacks increasingly specific to the memory condition under which they are generated. Attack effectiveness on the source condition increases, while proportional transfer retention decreases. Along individual optimization trajectories, gradients under the source and target memory conditions become less aligned, and perturbations optimized for different memory conditions become less similar. We refer to this process as *memory condition specialization*. Rather than simply producing a stronger version of the same transferable attack, optimization increasingly adapts the perturbation to the recurrent dynamics of the condition under which it is optimized. Our main findings are:

- **Write and erase play asymmetric adversarial roles.** Across GDN2, Mamba-1, and Hymba, write or input control has a larger effect on adversarial damage than erase or decay control. In GDN2, this ordering is opposite to the clean ablation trend.
- **Changing memory dynamics changes which attacks are effective.** A fixed perturbation can become less effective after the memory condition changes, even though an attack optimized for the new condition remains just as effective. In the GDN2 post attack experiment, approximately 95% of the apparent recovery is explained by attack mismatch.
- **Attack specialization grows during optimization.** Deeper optimization increases source attack effectiveness while reducing proportional transfer retention across memory conditions. This decline in retention is accompanied by lower gradient alignment and lower similarity between attacks optimized for different conditions.

2 Experiments

2.1 RQ1: Do write and erase have comparable effects on adversarial damage, or does one dominate?

Architectural premise: erase and write contribute differently to clean performance. Gated DeltaNet-2 (GDN2) separates the active memory edit into a key-side erase gate b_t and a value-side write gate w_t (Hatamizadeh et al., 2026). Its clean ablations reveal an asymmetry between the two controls. However, this clean-performance asymmetry does not determine their adversarial roles. We therefore test whether write or erase has a larger effect on adaptive attack damage and on a fixed attack downstream propagation.

Attack and statistical protocol. In our experiments, each evaluation example is processed as a fixed 576-token sequence, partitioned into 384 context tokens (positions 0–383), a 64-token attack span (positions 384–447), and a 128-token downstream evaluation span (positions 448–575). The context tokens establish the recurrent state before the attack and are never perturbed. Adversarial embedding perturbations are restricted to the attack span. Unless otherwise stated, gate interventions

are applied in all 18 recurrent layers over the same attack span.

$$\underbrace{\text{Context}}_{0-383} \rightarrow \underbrace{\text{Attack}}_{384-447} \rightarrow \underbrace{\text{Downstream evaluation}}_{448-575}$$

We multiply the post-sigmoid native gates at inference time, without modifying model weights:

$$w'_t = \lambda_w w_t, \qquad b'_t = \lambda_e b_t, \qquad \lambda_w, \lambda_e \in \{0.75, 1.00, 1.25\}. \tag{2}$$

We refer to $\lambda_w = \lambda_e = 1.00$ as the *Normal* setting where no intervention is applied. When varying write while keeping erase fixed at $\lambda_e = 1.00$, $\lambda_w = 0.75$ and 1.25 are denoted *Write-low* and *Write-high*, respectively. When varying erase while keeping write fixed at $\lambda_w = 1.00$, $\lambda_e = 0.75$ and 1.25 are denoted *Erase-low* and *Erase-high*.

We measure downstream adversarial damage over tokens 448–575 using the attack gap

$$A(c) = L_{\text{adv}}(c) - L_{\text{clean}}(c), \tag{3}$$

where $L_{\text{adv}}(c)$ and $L_{\text{clean}}(c)$ denote the mean next-token cross-entropy losses over the 128-token evaluation span under adversarial and clean inputs, respectively. Each intervention condition c is compared with its own clean baseline.

To determine which memory control has the larger effect on adaptive adversarial sensitivity, we compare write and erase under the same attack protocol. We use 128 C4 documents (Raffel et al., 2020) as the independent statistical units. Each erase or write condition is attacked separately using white-box APGD (Croce & Hein, 2020) with 40 optimization steps, cross-entropy and margin objectives, and three restarts per objective. Perturbations are constrained in embedding space by an L_∞ budget of $\epsilon = 0.00225$.

For each document and condition, the two objectives and three restarts produce six candidate attacks. We retain the valid candidate that produces the largest downstream evaluation loss. Using the same attack budget, optimizer, objectives, and restart count for both write and erase allows their adaptive effects to be compared directly.

Uncertainty is estimated using paired bootstrap confidence intervals over documents with 20,000 resamples. Tokens, objectives, restarts, and gate conditions are not treated as independent samples. All evaluated gate settings satisfy the prespecified clean performance guard of less than 5% relative change in clean loss and less than two percentage points of clean accuracy loss. The observed changes are substantially smaller than these limits.

Write dominates condition-specific adaptive sensitivity. We find that increasing write strength produces a large increase in attack damage under condition-specific optimization. The Write-high minus Write-low attack-gap difference is $+7.004$ (95% CI $[5.882, 8.166]$), with 127 of 128 documents showing the same positive direction (Figure 1). Increasing erase strength shifts damage in the opposite direction, but with a substantially smaller effect: the Erase-high minus Erase-low difference is -1.075 (95% CI $[-1.781, -0.390]$), with 81 of 128 documents showing reduced damage. The magnitude of the erase effect is approximately 15% of the write effect. Under the same model, documents, optimizer, objective suite, restart count, and perturbation budget, write therefore has a substantially larger effect on adaptive adversarial sensitivity, whereas stronger erase provides a smaller and less consistent protective effect.

Stronger erase partially counteracts write induced adversarial damage. We test whether increasing erase can reduce downstream attack damage while keeping both the adversarial perturbation and the write setting fixed. We perform two matched comparisons. First, we optimize a perturbation under Write-normal and Erase-normal, then replay the same perturbation under Write-normal and Erase-high. Second, we optimize a separate perturbation under Write-high and Erase-normal, then replay it under Write-high and Erase-high. In both comparisons, the write setting and adversarial perturbation are unchanged during replay, while only erase is increased during the attack span. After the attack span, both write and erase return to their normal settings.

For the same adversarial perturbation, increasing erase strength consistently reduces its downstream effect under both normal and strengthened write. The reduction is smaller under Write-high than under Write-normal (1.072 vs. 2.013). This shows that stronger erase can partially counteract the downstream influence of adversarial information written during the attack span, although this effect is weaker when write is strengthened.

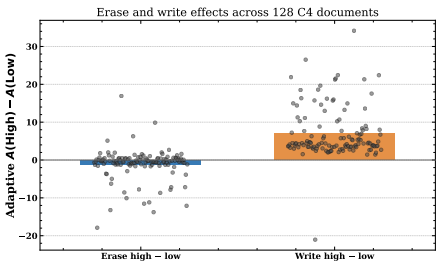

Figure 1: Write and erase have asymmetric effects on adaptive adversarial sensitivity. Points show paired document-level High–Low attack-gap differences under condition-specific APGD, and bars show means. Increasing write raises attack damage ($+7.004$), whereas increasing erase produces a smaller protective effect (-1.075).

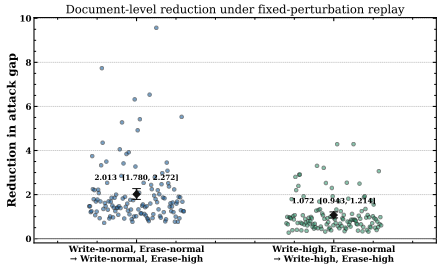

Figure 2: Increasing erase strength reduces the downstream effect of a fixed adversarial perturbation. Points show paired document-level reductions in attack gap from Erase-normal to Erase-high, and diamonds show means with 95% bootstrap confidence intervals. The reduction is smaller under Write-high than under Write-normal.

The write and erase asymmetry extends beyond GDN2. To test whether the adversarial asymmetry is specific to GDN2's explicit gates, we repeat a matched comparison within each model using Mamba-1 (Gu & Dao, 2024) and the recurrent attention hybrid Hymba (Dong et al., 2024). We use the public pretrained checkpoints `state-spaces/mamba-1.4b-hf` (State Spaces, 2024) and `nvidia/Hymba-1.5B-Base` (NVIDIA, 2024).

In GDN2, the erase gate is distinct from a separate decay operator. Decay broadly reduces the existing state, whereas erase selectively removes memory along the current key direction. Mamba-1 and Hymba do not provide an analogous independent erase gate, so we compare matched functional controls for new information input and existing state decay.

For both models, the relevant state update can be written schematically as

$$h_t = \underbrace{\exp(d_t A) h_{t-1}}_{\text{existing state}} + \underbrace{d_t B_t x_t}_{\text{new information}}, \tag{4}$$

where $A < 0$. Because d_t affects both old state decay and current input write, directly scaling d_t would change both operations. We therefore modify d_t and B_t jointly to separate these two effects.

For write, we keep the effective step size fixed and scale B_t:

$$d'_t = d_t, \qquad B'_t = \lambda_w B_t. \tag{5}$$

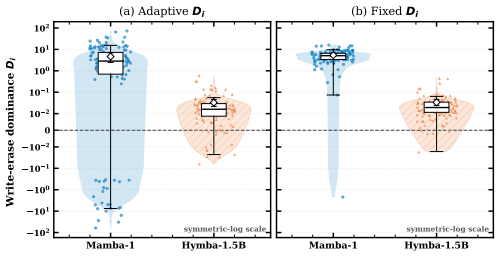

Figure 3: Input control has a larger adversarial effect than decay control in Mamba-1 and Hymba. Positive D_i indicates a larger input effect. The ordering holds under both condition specific adaptive attacks and fixed Normal source replay.

For old state decay, we scale d_t while compensating B_t:

$$d'_t = \lambda_e d_t, \qquad B'_t = \frac{1}{\lambda_e} B_t. \tag{6}$$

This keeps the instantaneous write contribution unchanged,

$$d'_t B'_t x_t = d_t B_t x_t, \tag{7}$$

while changing the old state transition from $\exp(d_t A)$ to $\exp(\lambda_e d_t A)$. In Hymba, the intervention is applied only to the SSM branch, while the parallel attention branch remains unchanged.

Both models are evaluated on the same 128 frozen C4 documents using APGD-40 with two objectives and three restarts per objective. Perturbation budgets are calibrated separately for each architecture.

For each model, we compare the magnitudes of the write and erase effects at the document level:

$$D_i = |\Delta_{W,i}| - |\Delta_{E,i}|,$$

where $\Delta_{W,i} = A_i(\text{Write-high}) - A_i(\text{Write-low})$ and $\Delta_{E,i} = A_i(\text{Erase-high}) - A_i(\text{Erase-low})$. Positive D_i indicates a larger write effect than erase for document i.

Mamba-1 and Hymba both reproduce the GDN2 ordering. In both architectures, input modulation has a larger effect on adversarial damage than erase modulation under condition specific adaptive attacks, and the same ordering remains when a Normal source perturbation is replayed unchanged across the write and erase conditions (Figure 3).

RQ1 conclusion. Across the three tested architectures, write or input control has a larger adversarial effect than erase or decay control. In GDN2, stronger erase can partially reduce the downstream influence of a fixed adversarial perturbation.

2.2 RQ2: Reduced Vulnerability or Attack Mismatch?

Post-attack erase produces small recovery of fixed adversarial damage. The fixed replay experiment in RQ1 changes erase during the attack span, testing whether erase alters the downstream propagation of adversarial information as it enters recurrent memory. We next ask a different question: once the attack span has ended, can increasing erase strength reduce adversarial damage that has already been written into the recurrent state? To isolate this post-attack effect, the Write-high intervention is confined to the attack span, tokens 384–447, while erase is increased from its normal setting ($\lambda_e = 1.00$) to Erase-high ($\lambda_e = 1.25$) only during the downstream evaluation span, tokens 448–575. Under the fixed Write-high perturbation, post attack Erase-high produces a small downstream recovery that increases with distance from the attack span, from 0.143 to 0.397 (Figure 4).

Adaptive evaluation reveals that fixed attack recovery is mostly attack mismatch. The recovery observed under a fixed perturbation can have two possible explanations. Post attack Erase-high may genuinely reduce adaptive attack damage, or the original perturbation may simply become less effective after the memory condition changes. To distinguish these possibilities, we compare two memory conditions:

Condition	Attack span (384–447)	Evaluation span (448–575)
W	Write-high, Erase-normal	Write-normal, Erase-normal
R	Write-high, Erase-normal	Write-normal, Erase-high

For each document, we optimize one perturbation under W and one under R. We then replay both perturbations on both conditions and, for each condition, retain the candidate that produces the larger

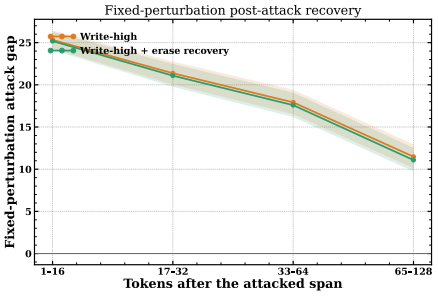

Figure 4: Fixed perturbation post attack recovery. Attack gap across downstream distance bins for Write-high with and without post attack Erase-high. Shaded regions show 95% bootstrap confidence intervals.

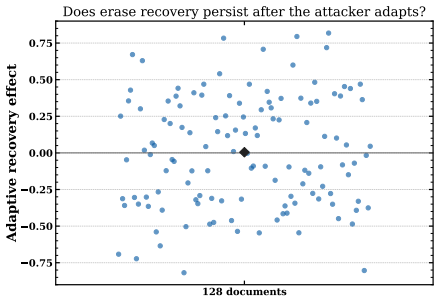

Figure 5: Post attack recovery under adaptive evaluation. Points show document level recovery effects, and the diamond marks the mean =0.0046 (95% CI $[-0.0608, 0.0706]$).

attack gap. Under this adaptive evaluation, the recovery is nearly zero: the mean effect is 0.0046, with 63 of 128 documents positive and 65 negative (Figure 5).

The contrast with fixed replay shows that the original perturbation can lose effectiveness after the memory condition changes without making the modified condition more robust, because an attack adapted to the new condition recovers almost all of the lost damage. We refer to this loss of transfer as *attack mismatch*.

Let $A_W(\delta)$ and $A_R(\delta)$ denote the attack gaps produced by perturbation δ under W and R, respectively. From the same two-candidate set, let $\hat{\delta}_W$ and $\hat{\delta}_R$ denote the candidates selected as strongest on W and R. The apparent fixed attack reduction can then be decomposed as

$$\underbrace{A_W(\hat{\delta}_W) - A_R(\hat{\delta}_W)}_{M_{\text{apparent}}} = \underbrace{A_W(\hat{\delta}_W) - A_R(\hat{\delta}_R)}_{M_{\text{adaptive}}} + \underbrace{A_R(\hat{\delta}_R) - A_R(\hat{\delta}_W)}_{M_{\text{mismatch}}}. \tag{8}$$

M_{adaptive} measures the change in adapted attack damage between the two memory conditions, whereas M_{mismatch} measures the loss from using the candidate selected for W on R. Because $\hat{\delta}_R$ is selected as the stronger candidate on R from the same candidate set, $M_{\text{mismatch}} \geq 0$.

Most apparent recovery is attack mismatch. Across the 128 C4 documents, the apparent recovery is 0.0850, while the adaptive component is only 0.0046. Attack mismatch accounts for the remaining 0.0804, or approximately 95% of the apparent recovery (Figure 6). Thus, most of the apparent recovery reflects the fixed attack losing effectiveness after the memory condition changes, rather than a measurable increase in adaptive robustness.

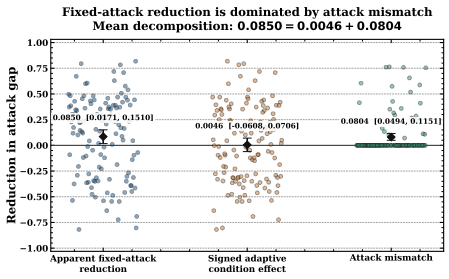

Figure 6: Decomposition of apparent recovery across 128 C4 documents. Points show document level values and diamonds show means with 95% paired bootstrap confidence intervals.

RQ2 conclusion. RQ2 identifies attack mismatch at a fixed optimization endpoint: a perturbation selected for one memory condition can lose effectiveness when replayed under another. RQ3 asks how this behavior changes with optimization depth. Specifically, does deeper condition specific optimization make attacks more effective on their source condition while reducing proportional transfer retention to other memory conditions? We refer to this pattern as memory condition specialization, where an attack becomes increasingly specific to the memory condition under which it is optimized.

2.3 RQ3: Does Optimization Produce Memory Condition Specialization?

From attack mismatch to memory condition specialization. We study this process across GDN2, Mamba-1, and Hymba using the same three write or input conditions defined in RQ1. For GDN2, the conditions are $\lambda_w \in \{0.75, 1.00, 1.25\}$; for Mamba-1 and Hymba, they are $\lambda_B \in \{0.75, 1.00, 1.25\}$, with erase or decay held at its normal setting. Each condition serves in turn as the source condition under which the attack is optimized, while the other two serve as transfer targets.

Optimization depth protocol and metrics. For each model, we use the same 16 frozen C4 documents and independently optimize APGD attacks for $K \in \{1, 10, 40, 80, 160\}$ steps. Each value of K is evaluated using a separately launched attack rather than a checkpoint from a longer trajectory. The perturbation budget, objectives, three restarts per objective, random start rule, projection procedure, and candidate selection rule are held fixed across optimization depths. We run three independent attack searches for each document, source condition, and value of K.

We measure attack effectiveness on the source condition using the clean correct flip rate, defined as the fraction of initially correct predictions that become incorrect under attack.

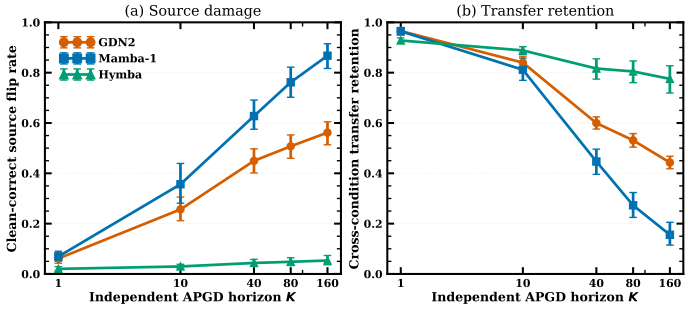

Figure 7: Progressive memory condition specialization across GDN2, Mamba-1, and Hymba. (a) Clean correct source flip rate increases and (b) transfer retention decreases across independently optimized APGD horizons K.

For document i, source condition s, target condition t, optimization depth K, and repeat r, let $A_{i,s\to t,K,r}$ denote the attack gap when a perturbation optimized under s is evaluated under t. We define transfer retention as

$$R_{i,s,K,r} = \frac{\frac{1}{2}\sum_{t\neq s} A_{i,s\to t,K,r}}{A_{i,s\to s,K,r}}. \tag{9}$$

Transfer retention measures the fraction of source condition attack damage retained after transfer to the other two memory conditions. Lower values indicate greater specialization to the source condition. This is a proportional measure. Absolute target attack gaps can still increase with optimization when source damage grows faster, so a decline in retention should not be read as a decline in absolute target damage.

Attack geometry changes together with specialization. We next examine how attack geometry changes as proportional transfer retention falls. For each attack, we run a single APGD-160 trajectory and record the best perturbation found at checkpoints $k \in \{1, 10, 40, 80, 160\}$. Unlike the independent optimization depth experiment above, these checkpoints come from the same trajectory.

For source condition s and target condition t, we measure gradient alignment and perturbation similarity as

$$\begin{aligned} G_{s\to t,k} &= \cos\left(\nabla_\delta L_s^{\mathrm{CE}}(x+\delta_{s,k}), \nabla_\delta L_t^{\mathrm{CE}}(x+\delta_{s,k})\right), \\ P_{s,t,k} &= \cos\left(\delta_{s,k}, \delta_{t,k}\right). \end{aligned} \tag{10}$$

Here, L_c^{CE} denotes the downstream cross entropy loss under memory condition c. Both gradients are recomputed at the same perturbed input using this common CE loss, independent of whether the corresponding APGD trajectory was optimized with the cross entropy or margin objective. Lower G indicates less similar local attack directions across memory conditions, while lower P indicates less similar perturbations optimized for different conditions. We track both quantities together with transfer retention along the same trajectory.

For this trajectory analysis, the perturbation budget is fixed within each model at $1.5\times$ the baseline budget for GDN2, $1.0\times$ for Mamba-1, and the calibrated $2.4\times$ budget for Hymba. These budgets provide sufficient source condition attack strength for trajectory analysis and should not be compared directly across models.

Across all three architectures, gradient alignment, perturbation similarity, and transfer retention decrease from checkpoint 1 to 160 (Figure 8). Gradient alignment decreases by 0.719, 0.724, and 0.571 for GDN2, Mamba-1, and Hymba, respectively; perturbation similarity decreases by 0.388, 0.390, and 0.500; and transfer retention decreases by 0.514, 0.646, and 0.407. The corresponding 95% document bootstrap confidence intervals exclude zero.

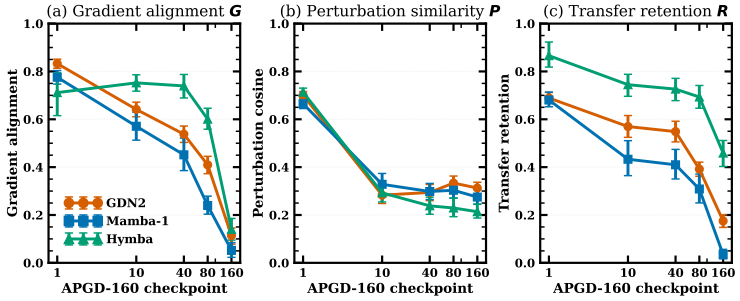

Figure 8: Attack geometry changes with memory condition specialization along single APGD-160 trajectories. (a) Same point CE gradient alignment, (b) perturbation similarity across memory conditions, and (c) transfer retention. Points show means over 16 C4 documents with 95% document bootstrap confidence intervals. All three metrics are lower at checkpoint 160 than at checkpoint 1 in GDN2, Mamba-1, and Hymba.

Together, these changes are consistent with progressive specialization to the source memory condition: local attack directions become less aligned across conditions, perturbations optimized under different conditions become less similar, and a smaller fraction of source condition attack damage remains after transfer.

RQ3 conclusion. Across the tested GDN2, Mamba-1, and Hymba checkpoints, greater optimization depth is associated with stronger attack effects on the source condition and lower proportional transfer retention to other memory conditions. Along single APGD-160 trajectories, this decline in retention is accompanied by decreasing same point gradient alignment and decreasing similarity between perturbations optimized under different memory conditions. Together, these results support progressive memory condition specialization: optimization increasingly adapts perturbations to the recurrent dynamics of the condition under which they are generated.

This specialization provides an explanation for the attack mismatch observed in RQ2. As optimization proceeds, attacks become increasingly condition specific, so a perturbation that is effective on its source condition can become less effective when replayed under another memory condition.

3 Discussion

Write and erase play asymmetric adversarial roles. Across GDN2, Mamba-1, and Hymba, write or input control has a larger effect on adversarial damage than erase or decay. GDN2 is also asymmetric under clean evaluation, but in the opposite direction, where erase control contributes more strongly than write control. The memory operation most important for clean computation therefore need not be the operation that creates the largest adversarial attack surface.

A possible explanation is that clean computation and adversarial attack rely on different memory functions. Erase helps manage previously stored information during normal processing, while write directly controls how strongly perturbed input enters memory during an attack. This may explain why erase contributes more to clean performance, whereas write has the larger effect on adversarial damage. This suggests a broader possibility: inputs that naturally induce stronger write activity may create memory regimes in which adversarial perturbations have greater downstream influence.

Fixed attack reduction should be interpreted as a transfer effect, not direct evidence of robustness. Changing recurrent memory dynamics changes not only the amount of adversarial damage but also which perturbations remain effective. Consequently, replaying a fixed perturbation after changing the memory condition mixes two effects: a change in the sensitivity of the target condition and a change in how well the original perturbation transfers to that condition. A reduction under fixed replay can therefore occur even when the modified condition remains similarly vulnerable to an adapted attack. Robustness claims under changing memory dynamics should accordingly rely on target adapted evaluation rather than fixed replay alone.

Optimization reveals condition specific adversarial structure. Our results suggest that the most effective attack strategy depends on the recurrent memory regime in which the attack is optimized. As optimization proceeds, perturbations become more effective on their original memory condition but retain a smaller fraction of source damage after transfer to other conditions, while gradients and optimized perturbations across conditions become less similar. This proportional decline does not imply that absolute target damage decreases. This raises the possibility that changes in context or prompting, which can alter the model's internal state and memory activity, may also change which adversarial perturbations are most effective. In this view, a prompt and an adversarial perturbation need not act independently: the prompt may first place the model in a different memory regime, thereby changing the attack directions favored by subsequent optimization. Our experiments motivate studying attacks that jointly account for context and recurrent state rather than treating the perturbation as independent of the model's current memory regime.

The shared pattern differs in magnitude across architectures. GDN2, Mamba-1, and Hymba share the same qualitative specialization pattern despite implementing recurrent memory differently. This agreement suggests that condition specific adversarial sensitivity is not limited to GDN2's explicit write and erase gates. Hymba shows smaller changes in some specialization metrics. One possible reason is that our intervention modifies only Hymba's SSM branch, while its parallel attention branch remains unchanged. Changes to SSM memory control therefore affect only part of the hybrid computation, which may reduce the separation between memory conditions at the model output. The magnitude is also attack strength dependent: Hymba shows substantially stronger trajectory separation under the moderate damage regime than under the baseline regime. These quantitative differences should not be interpreted as a robustness ranking because the models differ in architecture, checkpoints, intervention scope, and attack budgets.

4 Related Work

A growing body of work has studied adversarial robustness in state-space and recurrent models, including adversarial training and architecture-level robustness in deep SSMs (Qi et al., 2024), hidden-state poisoning in Mamba-based language models (Le Mercier et al., 2026), and control-theoretic analyses of adversarial vulnerability in SSM forecasting (Anand & Pappas, 2026). A separate line of work has examined adversarial transferability and the tendency of iterative optimization to over-specialize perturbations to source-specific gradient structure (Dong et al., 2018; Xie et al., 2019; Wang & He, 2021; Mi et al., 2026; Gao et al., 2026). Rather than studying transfer across different models, our work intervenes on recurrent memory controls within a fixed model and examines how these controls shape both adversarial sensitivity and attack specialization.

5 Conclusion

We study how recurrent memory dynamics shape adversarial sensitivity and attack specialization in GDN2, Mamba-1, and Hymba. Across the tested architectures, write or input control has a larger effect on adversarial damage than erase or decay control. In GDN2, this ordering reverses under clean ablations, where erase contributes more strongly to normal performance.

Changing memory dynamics also changes which perturbations remain effective. In GDN2, most of the apparent post attack recovery comes from the fixed attack losing effectiveness after the memory condition changes, rather than from lower vulnerability to an adapted attack. Across all three models, deeper optimization increasingly specializes attacks to their source memory condition, making them more effective on the source condition while reducing proportional transfer retention to other conditions.

These results show that recurrent memory shapes both the magnitude and specificity of adversarial vulnerability. When recurrent memory dynamics change, robustness should therefore be evaluated with attacks adapted to the modified condition rather than inferred from fixed attack replay alone.

Our results also suggest a possible connection to prompt based attacks. The model does not write every input into memory equally strongly, and different prompts may therefore create different conditions for an attack. In our experiments, manually increasing write strength made adversarial perturbations more damaging. This raises the possibility that an attacker could first choose a prompt

that naturally causes stronger memory updates, and then optimize a perturbation for that state. In this case, the prompt would help create a more favorable condition for the attack. Our results motivate studying prompt and perturbation design together.

REPRODUCIBILITY STATEMENT

To support reproducibility, our code and model checkpoints are available at `https://anonymous.4open.science/r/AR_recurrent-FEB5/README.md`.

AI USE STATEMENT

In this work, we used generative AI tools for generating code and text draft. We have not used generative AI tools for ideas. We have reviewed all AI-assisted work. We verified and tested LLM-generated code, we also verified the accuracy of the generated drafts and manually edited them sentence by sentence. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Sribalaji C. Anand and George J. Pappas. Adversarial robustness of deep state space models for forecasting, 2026. URL https://arxiv.org/abs/2604.03427.

Davi Bonetto. Spectralguard: Detecting memory collapse attacks in state space models. *arXiv preprint arXiv:2603.12414*, 2026.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216, 2020. URL https://proceedings.mlr.press/v119/croce20b.html.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models, 2024. URL https://arxiv.org/abs/2411.13676.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, 2018. doi: 10.1109/CVPR.2018.00957. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Dong_Boosting_Adversarial_Attacks_CVPR_2018_paper.html.

Xining Gao, Sen Huangfu, Ying Zhao, and Zijiao Zhang. Boosting targeted adversarial transferability via fine-grained feature mixup perturbation and reference-based gradient refinement. *Scientific Reports*, 16:23211, 2026. doi: 10.1038/s41598-026-57336-1. URL https://www.nature.com/articles/s41598-026-57336-1.

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

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: Decoupling erase and write in linear attention, 2026. URL https://arxiv.org/abs/2605.22791.

Alexandre Le Mercier, Chris Develder, and Thomas Demeester. Hidden state poisoning attacks against mamba-based language models, 2026. URL https://arxiv.org/abs/2601.01972.

Jian-Xun Mi, Xuanhui Zhong, and Weisheng Li. Improving adversarial transferability with local perturbation augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20639–20649, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Mi_Improving_Adversarial_Transferability_with_Local_Perturbation_Augmentation_CVPR_2026_paper.html.

NVIDIA. Hymba-1.5b-base pretrained checkpoint. Hugging Face model repository, 2024. URL https://huggingface.co/nvidia/Hymba-1.5B-Base. Repository: nvidia/Hymba-1.5B-Base; accessed 2026-09-02.

Biqing Qi, Yiang Luo, Junqi Gao, Pengfei Li, Kai Tian, Zhiyuan Ma, and Bowen Zhou. Exploring adversarial robustness of deep state space models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 6549–6573, 2024. doi: 10.52202/079017-0210. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0c8bbdf2657b58fa0a620f650fbdd457-Abstract-Conference.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

State Spaces. Mamba-1.4b-hf pretrained checkpoint. Hugging Face model repository, 2024. URL https://huggingface.co/state-spaces/mamba-1.4b-hf. Repository: state-spaces/mamba-1.4b-hf; accessed 2026-09-02.

Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1924–1933, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Enhancing_the_Transferability_of_Adversarial_Attacks_Through_Variance_Tuning_CVPR_2021_paper.html.

Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2730–2739, 2019. doi: 10.1109/CVPR.2019.00284. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Xie_Improving_Transferability_of_Adversarial_Examples_With_Input_Diversity_CVPR_2019_paper.html.

A Experimental Setup and Implementation Details

A.1 Models and memory interventions

Table 1 lists the three checkpoints. GDN2 uses `model-100b.pth` from the listed model repository. We apply each intervention at inference time and do not change model weights. The scale values are 0.75, 1.00, and 1.25. We call them Low, Normal, and High. Normal leaves the native computation unchanged.

Table 1: Models and intervention sites.

Model	Checkpoint	Layers	Intervention site
GDN2	`LLM-OS-Models2/gdn2-1.3b-paper-matched`	18	post sigmoid b_t, w_t
Mamba-1	`state-spaces/mamba-1.4b-hf`	48	selective scan d_t, B_t
Hymba	`nvidia/Hymba-1.5B-Base`	32	SSM branch d_t, B_t

For GDN2, the controller scales the post sigmoid erase gate b_t and write gate w_t passed to both the chunked and recurrent GDN2 kernels. A write intervention uses $w'_t = \lambda_w w_t$. An erase intervention uses $b'_t = \lambda_e b_t$. Unless a section states otherwise, the controller is active only at document positions 384 through 447 and in all 18 recurrent layers.

Mamba-1 and Hymba do not have GDN2's separate erase gate. We therefore use functional controls for new information input and old state decay. Their selective state update can be written as

$$h_t = \exp(d_t A) h_{t-1} + d_t B_t x_t, \qquad A < 0. \tag{11}$$

For input control, we keep d_t fixed and set $B'_t = \lambda_w B_t$. For decay control, we set

$$d'_t = \lambda_e d_t, \qquad B'_t = B_t / \lambda_e. \tag{12}$$

The product $d'_t B'_t$ then equals $d_t B_t$, while the old state transition changes to $\exp(\lambda_e d_t A)$. This compensation matches the implemented selective scan, which computes the old state factor as $\exp(d_t A)$ and the input contribution as $d_t B_t x_t$. It is not an approximation to a general full zero order hold expression for the discrete input matrix. In Mamba-1, $d_t = \mathrm{softplus}(\Delta_t + \mathrm{bias})$. We form the requested positive d'_t and map it back through the inverse softplus before the selective scan. The implementation checks that A is finite and negative. It leaves C_t, the skip term, the input, and all other tensors unchanged.

Hymba applies the same functional control only to the SSM branch. Its parallel attention branch is not changed. The input scale is applied after the branch's B normalization, because scaling before this normalization would be removed by scale invariance. Hymba prepends 128 internal memory tokens. Thus, external document positions 384 through 447 map to internal positions 512 through 575 at the intervention hook. Losses and all reported spans use the external document coordinates.

These controls match the function of new information input and old state decay, but they are not the same algebraic operation as GDN2's write and erase gates. We use "write or input" and "erase or decay" when a statement covers all three models.

Table 2 summarizes the terms used across models. "Write" and "erase" refer to the native GDN2 gates. "Input" and "decay" refer to the matched functions used in Mamba-1 and Hymba.

Table 2: Mapping between native quantities, interventions, and paper terms.

Model	Native quantity	Intervention	Paper term
GDN2	w_t	$w'_t = \lambda_w w_t$	write
GDN2	b_t	$b'_t = \lambda_e b_t$	erase
Mamba-1	B_t	$B'_t = \lambda_w B_t$	input
Mamba-1	d_t, B_t	$d'_t = \lambda_e d_t$, $B'_t = B_t / \lambda_e$	decay
Hymba	SSM B_t	$B'_t = \lambda_w B_t$ after normalization	input
Hymba	SSM d_t, B_t	compensated scaling as above	decay

A.2 Documents, tokenization, and spans

We use the validation split of C4 (Raffel et al., 2020), revision `f998d2cd8b92435980789e3ecb2f89b4c68bfe1e`. A frozen seed 3407 shuffle and manifest select 128 unique raw documents. We tokenize without adding special tokens and reject a document if it has fewer than 576 usable tokens. When at least 1025 tokens are available, we keep the centered 1025 token window. The model consumes the first 576 tokens of this stored window. For a shorter retokenized Mamba document, we keep all real tokens from the start and pad only the unused tail after them; the first 576 model inputs remain real document tokens. No token from another document is added. Each architecture tokenizes the same raw text with its own tokenizer, so token identities can differ across models even though the source document is shared.

Every example has 576 document tokens. Positions 0 through 383 form the clean context. Positions 384 through 447 form the 64 token attack span. Positions 448 through 575 form the 128 token downstream evaluation span. Only embeddings in the attack span may change. RQ1 and RQ2 use all 128 documents. RQ3 uses manifest indices 2 through 17, which gives 16 documents. This smaller set is a development cohort used for the larger optimization grid and is not a second held out confirmation.

Accuracy metrics use only target positions that are correct in the clean forward pass under the same condition. Documents with no clean correct target are rejected before aggregation. The saved manifests, token blocks, and span checks are fixed before attack selection.

A.3 Clean performance guard

Before interpreting an intervention, we require less than 5% relative increase in clean loss and less than two percentage points of clean accuracy loss compared with Normal. Table 3 gives the full RQ1 clean values. Every condition passes by a wide margin.

Table 3: Mean clean loss and next token accuracy on the 128 document RQ1 set.

Model	Condition	Clean loss	Clean accuracy
GDN2	Erase low	2.5679	0.4760
GDN2	Normal	2.5659	0.4760
GDN2	Erase high	2.5658	0.4756
GDN2	Write low	2.5663	0.4757
GDN2	Write high	2.5725	0.4747
GDN2	Both high	2.5709	0.4755
Mamba-1	Decay low	2.5227	0.4741
Mamba-1	Normal	2.5195	0.4733
Mamba-1	Decay high	2.5201	0.4727
Mamba-1	Input low	2.5238	0.4718
Mamba-1	Input high	2.5229	0.4727
Hymba	Decay low	2.1935	0.5233
Hymba	Normal	2.1920	0.5229
Hymba	Decay high	2.1924	0.5225
Hymba	Input low	2.1927	0.5219
Hymba	Input high	2.1929	0.5229

The run also checks that Normal matches the unhooked model, all intended layers are reached, the intervention is zero outside its planned span, and clean values agree across repeated paths within numerical tolerance.

For GDN2, the clean Write high minus Write low contrast is $+0.00614$ in loss and -0.00104 in accuracy, or -0.104 percentage points. The clean Erase high minus Erase low contrast is -0.00217 in loss and -0.00043 in accuracy, or -0.043 percentage points. These small changes describe our $\pm 25\%$ inference interventions. They are separate from the channel ablations reported for the original GDN2 model.

A.4 RQ1 FIXED REPLAY CONSTRUCTION

RQ1 uses two matched fixed replay comparisons. For the first, we optimize under Write normal and Erase normal, then replay the same perturbation under Write normal and Erase high. For the second, we optimize a separate perturbation under Write high and Erase normal, then replay it under Write high and Erase high. Within each comparison, the perturbation and write setting stay fixed. Only erase changes. Erase high is active only at positions 384 through 447. Both controls return to Normal after position 447. The two comparisons therefore use separately optimized perturbations and should not be read as four replays of one common perturbation.

A.5 NUMERICAL EXECUTION AND INTEGRITY CHECKS

Model inference uses the validated mixed precision path for each checkpoint. The optimized perturbation is stored in FP32 and is materialized with the model embeddings for each forward pass. We record both the requested FP32 budget and the realized change after materialization. Small differences caused by BF16 rounding are treated as numerical diagnostics, not as a larger optimization budget.

Every formal run checks the document manifest, token count, attack mask, condition name, layer coverage, finite losses, finite and nonzero gradients, nominal L_∞ budget, zero change outside the attack span, restart seeds, and replay identity. A candidate with a nonfinite loss or gradient, a wrong span, or a budget violation is invalid. Invalid candidates are not allowed to win. If a document has no valid candidate, the run fails instead of silently dropping that document. All reported aggregates passed these checks.

Table 4 lists the run information that is recorded consistently in the saved metadata. Software versions that were not stored for every run are not inferred.

Table 4: Implementation and execution summary.

Item	Recorded setting
Python	3.12.4
PyTorch and CUDA	2.9.0 with CUDA 12.8 in recorded attack runs
Transformers	4.57.0 in recorded Mamba runs
Precision	BF16 model forward; FP32 perturbation state
Accelerator	one NVIDIA A100 per recorded GDN2 and Mamba formal shard
Random seed	3407 base seed; deterministic offsets by document and restart

B Attack Optimization and Evaluation Metrics

B.1 Threat model and APGD procedure

Let $E(x)$ be the clean input embedding sequence and let M be the binary mask for positions 384 through 447. We optimize

$$E_\delta(x) = E(x) + M \odot \delta, \qquad \|\delta\|_\infty \le \epsilon. \tag{13}$$

The context and downstream evaluation embeddings are never changed. This is a white box continuous embedding attack. The exact baseline budgets are

$$\epsilon_{\mathrm{GDN2}} = 0.0022514138370752336, \quad \epsilon_{\mathrm{Mamba}} = 0.008462240696, \quad \epsilon_{\mathrm{Hymba}} = 0.0022514. \tag{14}$$

These are the exact RQ1 budgets. The RQ3 horizon experiment uses the values given in Section E. Budgets are set within each model and must not be compared as raw cross model robustness scores.

Budget selection. The base budget is defined within each checkpoint as

$$\epsilon_0 = 0.08\ \mathrm{RMS}(E), \tag{15}$$

where the RMS is computed from that checkpoint's embedding table. This gives 0.0015009425580501557 for GDN2, 0.008462240696 for Mamba-1, and 0.0025670528411865234 for Hymba. The common ratio normalizes for embedding scale, but it does not make the three threat models identical.

The GDN2 ratio was chosen in an earlier 16 document calibration as the smallest tested value that reached a target median downstream loss increase of about 10%. This threshold was close to a numerical boundary: two matched runs gave 9.84% and 10.02%. Later C4 experiments therefore fixed the value instead of applying the threshold again. The GDN2 RQ1 and RQ2 budget is $1.5\epsilon_0$. This stronger setting came from the earlier budget curve and was fixed before the additional 112 confirmation documents were evaluated. The first 16 development documents are included in the pooled set of 128, so this budget choice is not fully separate from the pooled analysis.

The Mamba-1 budget is its own $0.08\ \mathrm{RMS}(E)$ baseline. It was fixed in the Mamba pilot before the 128 document RQ1 run. Higher Mamba budgets approached prediction saturation, so the baseline was kept for RQ1 and the trajectory experiment. The Hymba RQ1 budget, 0.0022514, was fixed before its 128 document run. That protocol did not include an epsilon sweep or a multiplier calibration, so we do not describe this value as a damage matched budget.

The independent horizon experiment uses each model's ϵ_0 and does not select a budget from the five horizon results. For the single trajectory experiment, GDN2 uses $1.5\epsilon_0$ to provide a stronger source attack, and Mamba-1 uses $1.0\epsilon_0$ to avoid the higher budget saturation seen in the earlier budget curve. Hymba reports both $1.0\epsilon_0$ and a moderate setting. The moderate multiplier was calibrated only on manifest indices 0 and 1, which are separate from the formal indices 2 through 17. The rule selected a Normal condition clean correct flip rate in $[0.35, 0.65]$ without reading alignment, similarity, or transfer. The selected multiplier was 2.4, with calibration flip rate 0.483, and it was fixed before the formal trajectories.

These choices target useful, nonsaturated attack regimes within each model. They do not support a comparison of raw epsilon or raw attack gap across architectures. The models differ in embedding

scale, tokenizer, checkpoint, precision, and intervention, and the same relative budget can produce different clean correct flip rates.

Each restart begins from a uniform random point in $[-\epsilon, \epsilon]$ on the attack span. Paired conditions use the same restart seeds. The first step size is 2ϵ. Updates use the sign of the gradient, a momentum weight of 0.75 after the first update, and projection back to the L_∞ ball after every step. There is no image style $[0, 1]$ clamp.

The step size is halved when the objective oscillates or does not improve. The oscillation rule requires progress on at least 75% of updates in the current check window. After a reduction, APGD returns to the best saved perturbation and its gradient. The first check window is $\lfloor 0.22K \rfloor$, the minimum window is $\lfloor 0.06K \rfloor$, and the window is shortened by $\lfloor 0.03K \rfloor$ after each check, with integer guards for short runs. APGD-40 means 40 gradient updates and 41 objective evaluations. APGD-160 means 160 gradient updates and 161 objective evaluations.

We use two objectives. The CE objective is the mean next token cross entropy over the downstream evaluation span. The margin objective is

$$L^{\text{margin}} = \frac{1}{128} \sum_j \left[\max_{v \neq y_j} z_{j,v} - z_{j,y_j} \right], \tag{16}$$

where $z_{j,v}$ is the logit for vocabulary item v and y_j is the correct next token. We maximize both objectives. The step size checks and rollback follow the objective used by that trajectory. At every query, including the initial point, we also record the common downstream CE. The returned candidate is the point on that trajectory with the largest common CE, which can differ from the point with the largest margin. For each condition, we run three CE restarts and three margin restarts. This gives six returned candidates. We select the valid candidate with the largest common downstream CE. Thus, both the within trajectory and final candidate choices use the same evaluation loss.

B.2 Losses, attack gaps, and accuracy metrics

For condition c, the clean loss $L_{\text{clean},i}(c)$ and adversarial loss $L_{\text{adv},i}(c)$ are means over the 128 downstream targets of document i. The attack gap is

$$A_i(c) = L_{\text{adv},i}(c) - L_{\text{clean},i}(c). \tag{17}$$

Each condition uses its own clean loss. This removes the small clean shift caused by the intervention from the attack effect.

The clean correct flip rate is the number of clean correct downstream predictions that become wrong under attack, divided by the number of clean correct predictions. For a perturbation optimized under source condition s and replayed under target t, we write its attack gap as $A_{i,s\to t}$. RQ3 transfer retention is

$$R_{i,s} = \frac{\frac{1}{2} \sum_{t \neq s} A_{i,s\to t}}{A_{i,s\to s}}. \tag{18}$$

We first average the two off diagonal targets within a source, then average attack repeats and source conditions within a document, and finally average documents. A token, objective, restart, condition, or replay cell is never treated as an independent sample.

All source condition gaps used as denominators in the independent horizon experiment are positive. Across the 240 document, horizon, and source records for each model, the minimum denominator is 0.0719 for GDN2, 0.1209 for Mamba-1, and 0.0192 for Hymba. No retention value is negative. Retention can exceed one when the mean transfer gap is larger than the source gap, so it is not clipped. This occurs in 32 of 240 GDN2 records, 33 of 240 Mamba-1 records, and 47 of 240 Hymba records.

B.3 Statistical procedure

The document is the independent unit. All main contrasts are paired by document. We use a percentile paired bootstrap with seed 3407. The GDN2 RQ1 and RQ2 summaries use 20,000 resamples. The saved Mamba-1 and Hymba RQ1 summaries use 10,000 resamples. The independent horizon summaries use 10,000. The GDN2 and Mamba-1 trajectory summaries use 20,000, while the

Hymba trajectory summary uses 10,000. In each resample, complete document records are drawn with replacement, so all conditions, sources, targets, objectives, and repeats from a document remain paired. Reported intervals are two sided 95% intervals from the 2.5 and 97.5 percentiles.

For main paired effects, we report the mean, median, 10% trimmed mean, direction counts, and the range of leave one document out means. RQ3 fits, for each document,

$$R_{i,K} = \alpha_i + \beta_i \log_2(K+1) + \varepsilon_{i,K}, \tag{19}$$

and bootstraps the mean document slope. The tested contrasts were set by the research questions. We do not apply a multiple comparison correction to these planned contrasts. Extra cells and intervals are descriptive and are not treated as separate claims.

C Additional RQ1 Results

C.1 Full GDN2 contrasts

Table 5 reports the full robust summaries behind Section 2.1. Increasing write raises adaptive attack damage, while increasing erase has a smaller mean effect in the other direction. The result is not caused by one document: every leave one document out mean remains positive for write and negative for erase.

Table 5: GDN2 adaptive High minus Low contrasts. LOO gives the range of leave one document out means.

Control	Mean	Median	Trimmed	95% CI	Positive	LOO range
Write	7.004	4.414	6.055	[5.882, 8.166]	127/128	[6.790, 7.225]
Erase	−1.075	−0.359	−0.588	[−1.781, −0.390]	47/128	[−1.217, −0.943]

The selected mean attack gaps were 9.054 for Write low, 12.379 for Normal, 16.059 for Write high, 13.165 for Erase low, and 12.089 for Erase high. Their medians were 8.516, 10.764, 13.021, 10.791, and 10.684. These values use a separate adaptive attack for each condition.

C.2 Fixed perturbation replay results

The construction is given in Section A. For the Write normal comparison, the mean attack gap decreases from 8.303 to 6.290 when erase is increased, a reduction of 2.013 with 95% CI $[1.780, 2.272]$. For the Write high comparison, it decreases from 16.059 to 14.986, a reduction of 1.072 with 95% CI $[0.943, 1.214]$. All 128 documents have a positive reduction in both comparisons. The difference between the two reductions is 0.941, with median 0.719, 10% trimmed mean 0.782, 95% CI $[0.791, 1.109]$, and 124 of 128 documents positive. Its leave one document out means range from 0.907 to 0.953.

C.3 Mamba-1 and Hymba

For document i, let $\Delta_{W,i}$ be the Input high minus Input low attack gap and let $\Delta_{E,i}$ be the Decay high minus Decay low attack gap. We compare their magnitudes with

$$D_i = |\Delta_{W,i}| - |\Delta_{E,i}|. \tag{20}$$

Positive D_i means that input control has the larger effect. Adaptive attacks are optimized separately for each target condition. Fixed replay uses one Normal source perturbation and evaluates it unchanged under all target conditions.

Table 6: Cross architecture RQ1 contrasts. The last column counts documents with positive D.

Model and mode	Δ_W	95% CI	Δ_E	95% CI	D	Median D	95% CI for D	Positive D
Mamba-1 adaptive	8.822	[6.826, 10.980]	−1.761	[−3.349, −0.211]	4.667	2.849	[2.455, 6.910]	104/128
Mamba-1 fixed	6.474	[5.902, 7.079]	−0.435	[−0.743, −0.138]	5.357	5.093	[4.851, 5.889]	127/128
Hymba adaptive	0.0400	[0.0265, 0.0565]	−0.0048	[−0.0120, 0.0007]	0.0326	0.0160	[0.0217, 0.0460]	113/128
Hymba fixed	0.0440	[0.0332, 0.0572]	0.0025	[−0.0006, 0.0060]	0.0336	0.0193	[0.0243, 0.0452]	124/128

For Mamba-1, the adaptive write contrast has median 4.885 and is positive for 118 of 128 documents. The adaptive erase contrast has median -0.984 and is negative for 88 documents. Under fixed replay, the corresponding medians are 6.098 and -0.409, with 128 positive write effects and 85 negative erase effects. The 10% trimmed means of adaptive and fixed D are 3.965 and 5.116. Their leave one document out ranges are $[4.163, 5.174]$ and $[5.275, 5.415]$.

For Hymba, the adaptive mean absolute input and decay effects are 0.04698 and 0.01438. Under fixed replay, they are 0.04406 and 0.01046. The individual decay contrasts are not distinguishable from zero at the 95% level because their intervals include zero. The adaptive D count is 113 positive and 15 negative; the fixed count is 124 positive and 4 negative. Both paired D intervals remain above zero, supporting the within model conclusion that input control has the larger effect. Raw magnitudes should only be compared within a model because the models use different checkpoints, tokenizers, embedding scales, and budgets.

D Additional RQ2 Analysis

D.1 Post attack timing and distance

RQ2 uses two policies that are identical through the attack span:

	context $0{:}383$	attack $384{:}447$	evaluation $448{:}575$
W	Normal	Write high	Normal
R	Normal	Write high	Erase high

(21)

Thus, W and R have the same clean context and the same Write high attack span. They differ only after the attack span: R turns on Erase high during downstream evaluation. This timing separates the removal of already stored attack effects from the attack span erase test in RQ1.

For a fixed W source perturbation, Table 7 gives the attack gaps and recovery in each downstream bin. Recovery is the W gap minus the R gap.

Table 7: Fixed perturbation recovery by distance from the attack span. The full span row uses token weights.

Downstream tokens	W gap	R gap	Recovery [95% CI]
1–16	25.327	25.184	0.143 [0.108, 0.180]
17–32	21.371	21.083	0.288 [0.235, 0.343]
33–64	17.923	17.591	0.332 [0.284, 0.381]
65–128	11.481	11.084	0.397 [0.350, 0.444]
Full span	16.059	15.723	0.335 [0.303, 0.368]

The rows in Table 7 are mean per token gaps within bins of different lengths. We therefore aggregate the full span with token weights. For document i, let $n_b \in \{16, 16, 32, 64\}$ be the bin length. We compute

$$A^{\text{full}}_{W,i} = \frac{1}{128} \sum_b n_b A_{W,i,b}, \tag{22}$$

$$A^{\text{full}}_{R,i} = \frac{1}{128} \sum_b n_b A_{R,i,b}, \tag{23}$$

$$M^{\text{full}}_i = A^{\text{full}}_{W,i} - A^{\text{full}}_{R,i}, \tag{24}$$

$$q_i = \frac{M^{\text{full}}_i}{A^{\text{full}}_{W,i}}. \tag{25}$$

The reported recovery fraction is the mean of q_i across documents. It is not a ratio computed after averaging documents. These values were recomputed from the saved document and bin records; the older aggregate fields that directly summed the four bin means are not used.

The mean full span recovery is 0.3355, with 95% CI $[0.3025, 0.3679]$, median 0.3490, and 10% trimmed mean 0.3344. It is positive for 122 of 128 documents, and the leave one document out means range from 0.3317 to 0.3392. The mean document recovery fraction is 2.33%, with 95% CI $[2.10\%, 2.55\%]$, median 2.47%, and 10% trimmed mean 2.37%. Recovery grows with distance, but remains small compared with total attack damage.

D.2 Adaptive evaluation and mismatch decomposition

For each document, the search over six candidates under W first returns one source selected perturbation, δ^*_W. The same search under R returns δ^*_R. We replay both selected perturbations under both policies. This gives four cells: $\delta^*_W \to W$, $\delta^*_W \to R$, $\delta^*_R \to W$, and $\delta^*_R \to R$. Thus, adaptive evaluation is performed over the two source selected candidates

$$\mathcal{C}_i = \{\delta^*_{W,i}, \delta^*_{R,i}\}. \tag{26}$$

We select

$$\hat{\delta}_{W,i} = \arg\max_{\delta \in \mathcal{C}_i} A_{W,i}(\delta), \qquad \hat{\delta}_{R,i} = \arg\max_{\delta \in \mathcal{C}_i} A_{R,i}(\delta). \tag{27}$$

This procedure does not search again over all 12 original candidates. The target winners $\hat{\delta}_{W,i}$ and $\hat{\delta}_{R,i}$ are selected only from $\mathcal{C}_i$. The candidate selected for a target need not come from the search run under that target policy.

Let

$$M_{\text{apparent},i} = A_{W,i}(\hat{\delta}_{W,i}) - A_{R,i}(\hat{\delta}_{W,i}), \tag{28}$$

$$M_{\text{adaptive},i} = A_{W,i}(\hat{\delta}_{W,i}) - A_{R,i}(\hat{\delta}_{R,i}), \tag{29}$$

$$M_{\text{mismatch},i} = A_{R,i}(\hat{\delta}_{R,i}) - A_{R,i}(\hat{\delta}_{W,i}). \tag{30}$$

Adding the last two lines cancels $A_{R,i}(\hat{\delta}_{R,i})$ and gives

$$M_{\text{apparent},i} = M_{\text{adaptive},i} + M_{\text{mismatch},i}. \tag{31}$$

The identity holds exactly in the stored document records. Since $\hat{\delta}_{R,i}$ maximizes the R gap over the same two candidates, $M_{\text{mismatch},i} \geq 0$. A zero means that the candidate selected under W is also strongest under R.

Table 8 gives the complete component summaries. The adaptive attack gaps are 16.903 under W and 16.899 under R. Mismatch is zero for 102 documents because the same candidate wins under R, and positive for the other 26. The exact decomposition has maximum stored error zero.

Table 8: RQ2 decomposition over 128 documents. LOO is the range of leave one document out means.

Component	Mean	Median	Trimmed mean	95% CI	Negative / zero / positive	LOO
M_{apparent}	0.0850	0.1721	0.0983	[0.0171, 0.1510]	48 / 0 / 80	[0.0792, 0.0921]
M_{adaptive}	0.0046	−0.0069	0.0021	[−0.0608, 0.0706]	65 / 0 / 63	[−0.0018, 0.0111]
M_{mismatch}	0.0804	0	0.0294	[0.0494, 0.1151]	0 / 102 / 26	[0.0750, 0.0810]

The mean mismatch is about 95% of the mean apparent recovery. This figure is the ratio of the aggregate mean components, $0.0804/0.0850$. It is not the fraction of documents with nonzero mismatch. In fact, mismatch is nonzero for 26 of 128 documents. The fixed perturbation therefore loses transfer after the policy changes, while the best adapted damage is almost unchanged.

E Additional RQ3 Analysis

E.1 Independent optimization depth experiment

The independent horizon experiment uses $K \in \{1, 10, 40, 80, 160\}$. The grid includes a nearly unoptimized attack, the main 40 step setting, and longer 80 and 160 step searches, while keeping

the total cost manageable. Each K is a separately launched APGD search. It is not a saved point from the $K = 160$ run. Because the APGD schedule depends on K, the precise comparison is across optimization search configurations. We keep the model budget, objective set, restart rule, projection, and selection rule fixed. For every document, source condition, and K, we run three independent searches. The three write or input conditions rotate through the source role, and each selected perturbation is replayed under the other two conditions. This design measures the result of giving the optimizer a larger search horizon.

The exact independent horizon budgets are 0.0015009425580501557 for GDN2, 0.008462240695953369 for Mamba-1, and 0.0025670528411865234 for Hymba. They stay fixed across all five values of K.

Tables 9 and 10 report every K. Source gap is the mean diagonal attack gap, Flip is the clean correct source flip rate, and R is transfer retention.

Table 9: Independent horizon results for GDN2 and Mamba-1. Intervals are document bootstrap 95% intervals.

Model	K	Source gap	Flip	Flip 95% CI	R	R 95% CI
GDN2	1	0.206	0.061	[0.043, 0.085]	0.965	[0.951, 0.978]
GDN2	10	1.598	0.257	[0.212, 0.306]	0.840	[0.812, 0.866]
GDN2	40	5.363	0.450	[0.401, 0.498]	0.600	[0.576, 0.624]
GDN2	80	7.031	0.507	[0.459, 0.553]	0.531	[0.504, 0.558]
GDN2	160	9.071	0.562	[0.513, 0.605]	0.443	[0.418, 0.468]
Mamba-1	1	0.239	0.069	[0.051, 0.090]	0.965	[0.950, 0.980]
Mamba-1	10	2.107	0.357	[0.281, 0.439]	0.812	[0.769, 0.858]
Mamba-1	40	10.968	0.628	[0.575, 0.691]	0.447	[0.396, 0.496]
Mamba-1	80	23.197	0.762	[0.702, 0.822]	0.273	[0.225, 0.324]
Mamba-1	160	44.625	0.868	[0.817, 0.915]	0.156	[0.115, 0.205]

Table 10: Independent horizon results for Hymba.

K	Source gap	Flip	Flip 95% CI	R	R 95% CI
1	0.0648	0.0202	[0.0125, 0.0291]	0.9279	[0.9172, 0.9381]
10	0.1153	0.0298	[0.0211, 0.0384]	0.8884	[0.8730, 0.9030]
40	0.1642	0.0437	[0.0306, 0.0580]	0.8165	[0.7743, 0.8549]
80	0.1979	0.0483	[0.0335, 0.0645]	0.8049	[0.7602, 0.8466]
160	0.2428	0.0534	[0.0364, 0.0735]	0.7752	[0.7190, 0.8275]

The mean retention slopes against $\log_2(K + 1)$ are -0.0861 for GDN2, -0.1353 for Mamba-1, and -0.0249 for Hymba. Their 95% intervals are $[-0.0914, -0.0805]$, $[-0.1422, -0.1274]$, and $[-0.0335, -0.0169]$. All 16 document slopes are negative in every model. From $K = 1$ to $K = 160$, retention changes by -0.5226, -0.8088, and -0.1527, with intervals $[-0.5500, -0.4934]$, $[-0.8455, -0.7636]$, and $[-0.2057, -0.1047]$.

E.2 Single trajectory geometry experiment

This experiment asks a different question. It runs one APGD-160 trajectory and saves its best perturbation at steps 1, 10, 40, 80, and 160. The saved points therefore share one random start and all

earlier updates. For source s, target t, and saved perturbation $\delta_{s,k}$, we define

$$G_{s\to t,k} = \cos\big(\nabla_\delta L_s^{\mathrm{CE}}(x+\delta_{s,k}), \nabla_\delta L_t^{\mathrm{CE}}(x+\delta_{s,k})\big)\,, \tag{32}$$

$$P_{s,t,k} = \cos(\delta_{s,k}, \delta_{t,k}). \tag{33}$$

Both gradients in G are recomputed at the same perturbed input. They use the same downstream CE loss even when the saved trajectory was optimized with the margin objective. This same point CE rule was checked in the GDN2, Mamba-1, and Hymba trajectory runners. P compares perturbations from matched document, repeat, objective, and step records under two source conditions.

Table 11: Single trajectory results for GDN2 and Mamba-1. Each metric is mean [95% document bootstrap CI].

Model	Step	Gradient alignment G	Perturbation similarity P	Transfer retention R	Source flip rate
GDN2	1	.833 [.813, .851]	.701 [.685, .716]	.689 [.666, .712]	.093 [.067, .123]
GDN2	10	.641 [.611, .672]	.283 [.248, .324]	.570 [.525, .616]	.325 [.290, .358]
GDN2	40	.538 [.503, .572]	.294 [.260, .328]	.549 [.505, .592]	.440 [.398, .480]
GDN2	80	.409 [.373, .445]	.334 [.304, .363]	.392 [.362, .421]	.579 [.532, .621]
GDN2	160	.114 [.078, .146]	.313 [.288, .337]	.175 [.148, .201]	.857 [.820, .896]
Mamba-1	1	.776 [.747, .804]	.665 [.645, .683]	.681 [.652, .714]	.080 [.056, .108]
Mamba-1	10	.571 [.513, .630]	.328 [.287, .374]	.433 [.364, .511]	.393 [.265, .537]
Mamba-1	40	.452 [.386, .522]	.298 [.265, .332]	.410 [.351, .474]	.544 [.415, .677]
Mamba-1	80	.240 [.203, .279]	.303 [.270, .336]	.309 [.250, .371]	.717 [.612, .821]
Mamba-1	160	.052 [.023, .085]	.275 [.244, .306]	.035 [.017, .056]	.925 [.871, .971]

From step 1 to 160, G changes by -0.719 for GDN2, -0.724 for Mamba-1, and -0.571 for Hymba at the moderate budget. The 95% intervals are $[-0.761, -0.680]$, $[-0.774, -0.672]$, and $[-0.652, -0.484]$. The corresponding changes in P are -0.388, -0.390, and -0.500, with intervals $[-0.421, -0.355]$, $[-0.421, -0.360]$, and $[-0.529, -0.466]$. Changes in R are -0.514, -0.646, and -0.407, with intervals $[-0.544, -0.485]$, $[-0.687, -0.610]$, and $[-0.471, -0.342]$. All 16 documents have negative changes for these three metrics in the reported GDN2, Mamba-1, and moderate Hymba settings. The source flip rate increases by 0.763 for GDN2, 0.845 for Mamba-1, and 0.673 for moderate Hymba, with intervals $[0.712, 0.816]$, $[0.794, 0.892]$, and $[0.597, 0.748]$.

E.3 Trajectory budget sensitivity in Hymba

The trajectory budgets are model specific. GDN2 uses 1.5 times its horizon baseline, which gives $\epsilon = 0.0022514138370752336$. Mamba-1 uses its horizon baseline, $\epsilon = 0.008462240695953369$. Hymba is reported in the main text at a calibrated 2.4 times baseline budget. Hymba's trajectory baseline is $\epsilon = 0.0025670528411865234$. Its moderate damage budget is $\epsilon = 0.006160926818847656$. The multiplier was selected with separate calibration documents before the 16 formal documents were evaluated.

Table 12: Hymba trajectories under the baseline and moderate budgets. Each metric is mean [95% document bootstrap CI].

Budget	Step	Gradient alignment G	Perturbation similarity P	Transfer retention R	Source flip rate
Baseline	1	.887 [.865, .906]	.714 [.692, .731]	.786 [.750, .821]	.025 [.015, .037]
Baseline	10	.855 [.813, .890]	.628 [.592, .659]	.683 [.640, .725]	.035 [.023, .047]
Baseline	40	.844 [.792, .889]	.599 [.564, .635]	.629 [.583, .676]	.041 [.029, .054]
Baseline	80	.833 [.764, .887]	.593 [.560, .629]	.597 [.551, .648]	.047 [.032, .065]
Baseline	160	.811 [.724, .880]	.578 [.537, .618]	.566 [.508, .624]	.055 [.036, .076]
Moderate	1	.712 [.615, .794]	.714 [.692, .730]	.866 [.818, .923]	.039 [.029, .051]
Moderate	10	.752 [.718, .786]	.293 [.254, .330]	.745 [.696, .788]	.125 [.090, .165]
Moderate	40	.739 [.689, .787]	.238 [.203, .275]	.726 [.678, .771]	.264 [.198, .337]
Moderate	80	.600 [.559, .646]	.229 [.193, .271]	.693 [.646, .741]	.476 [.385, .567]
Moderate	160	.140 [.095, .186]	.214 [.187, .246]	.459 [.403, .511]	.712 [.636, .786]

For the baseline budget, the step 1 to 160 changes are -0.075 for G, -0.136 for P, -0.221 for R, and $+0.030$ for source flip rate. Their intervals are $[-0.149, -0.014]$, $[-0.170, -0.101]$, $[-0.279, -0.168]$, and $[0.017, 0.042]$. Ten documents have lower G, all 16 have lower P and R, and 15 have a higher flip rate. For the moderate budget, all 16 documents have lower G, P, and R, and all 16 have a higher source flip rate.

The stronger Hymba setting gives a larger quantitative specialization signal. It also causes much more damage: the mean source flip rate rises from 0.039 at step 1 to 0.712 at step 160. Under the baseline setting it rises only from 0.025 to 0.055. The moderate setting therefore supports the endpoint result, but it does not isolate a causal path from geometry to transfer because damage and possible saturation change at the same time. The baseline result shows the same direction with less damage, but with smaller changes.

F Limitations and Additional Discussion

The threat model uses continuous embedding perturbations. It does not produce discrete tokens or natural language prompts. A strong embedding attack can reveal local model sensitivity, but it need not map to a readable prompt with the same effect. Testing discrete search and natural prompt attacks is an important next step.

APGD searches a bounded space and does not prove a global worst case. More restarts, other objectives, or another optimizer may find a stronger candidate. Our use of two objectives, three restarts, best point tracking, and integrity checks reduces search error but cannot remove this limit.

The memory interventions are tools for analysis. They are not a complete defense and are not directly a deployment method. GDN2's write and erase gates, Mamba-1's input and decay controls, and Hymba's SSM branch controls are functionally matched but not algebraically identical. Hymba also keeps its attention branch unchanged. These facts limit direct claims about equal mechanisms across architectures.

Raw attack gap size is not a robustness ranking across models. The checkpoints use different tokenizers, embedding scales, architectures, intervention scopes, and perturbation budgets. Low transfer also does not mean low source vulnerability. An attack can have low transfer because it is highly adapted to one condition while still causing large damage there.

Our evidence covers one checkpoint from each model family, one C4 span layout, and the tested scale and budget ranges. It does not show that the same sizes hold for all recurrent language models, other data, other token positions, fine tuning, or quantization. Fine tuning and quantization can change both the clean model and its local attack geometry, so they are not equivalent to the inference controls used here.

The observation that stronger write or input control increases attack damage suggests a testable hypothesis: inputs that naturally produce larger write activity may offer a larger path for adversarial influence. We do not test natural prompts or claim that write activity alone causes vulnerability.

Direct measurement of native gate or input activity, followed by prompt level attacks, would be needed.

Finally, decreasing gradient alignment, perturbation similarity, and transfer retention gives a consistent picture of memory condition specialization, but this geometry is not a unique mechanism. Other changes in representation, curvature, or output sensitivity could produce the same pattern. The current results show that specialization develops with optimization under the tested memory conditions.