GENERALIZATION BEFORE MEMORIZATION: HOW EARLY OBJECTIVE PRIORITIZATION SHAPES SUBSEQUENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) both learn abstract patterns that generalize to unseen examples and memorize specific knowledge that cannot be inferred from patterns. Although models typically learn both objectives concurrently, we identify a fundamental learning order asymmetry: prioritizing generalization first leads to stronger subsequent generalization, whereas prioritizing memorization first can make generalization harder to acquire. Across 54 model–dataset configurations spanning six architectures and nine tasks, generalization-first training improves generalization performance in 74.1% of cases while preserving memorization. Conversely, a memorization-first regime frequently prevents models from developing skills beyond memorizing data. To further investigate this finding, we conduct an analysis of intermediate neural representations across training, and isolate specific sub-networks responsible for this behavioral divergence. Our findings demonstrate that training order governs optimization trajectories, highlighting the need to prioritize generalization early in LLM training schedules.

1 INTRODUCTION

Language models learn different capabilities during training, and these capabilities may be acquired at different stages. Some behaviors require learning patterns that transfer to unseen examples, while others require memorizing specific information that cannot be inferred from patterns or rules. If learning one behavior first changes how easily another can be learned later, then a model's final capabilities depend not only on what objectives it is trained on, but also on what was prioritized earlier. Understanding this effect is important for both explaining training outcomes and designing how multiple objectives are introduced during training.

In this work, we study two forms of learning in a controlled setting: : generalization and memorization. We use **generalization (GEN)** to refer to learning a structured task rule and applying it to held-out examples, and **memorization (MEM)** to refer to learning fixed prompt–target associations that cannot be derived from that rule. A model may ultimately need to learn both, raising the question of how learning one affects the later acquisition of the other.

Prior work suggests that early training can have lasting effects. Curriculum learning shows that changing which examples are presented first can alter optimization and generalization [2, 14, 35], while grokking and related studies show that memorization and generalization can emerge differently over training [26, 21]. However, these results do not directly answer whether what a model is trained to learn first still matters after both behaviors become training objectives. If later joint optimization removes the effect of early learning, different initial priorities should lead to similar outcomes. If it does not, early objective prioritization may remain consequential even after the objectives become shared.

We test this directly by controlling which behavior is prioritized first. In **GenMix**, models first learn GEN and then transition to joint GEN+MEM training; in **MemMix**, they first learn MEM and then transition to the same joint training procedure. We additionally use **PureMix**, which trains on GEN and MEM jointly from the start, as a reference showing whether the two behaviors can coexist. This design isolates early objective prioritization while keeping the subsequent GenMix and MemMix objective identical.

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.

Across 54 model–dataset settings spanning six models and nine tasks, we find a clear asymmetry. GenMix achieves higher final held-out GEN than MemMix in 40 of 54 settings (74.1%), with mean GEN of 44.49% versus 32.47%. At the same time, PureMix shows that strong GEN and MEM can coexist, suggesting that the lower GEN after MEM-first training is not simply due to incompatibility between the two objectives. To better understand why GEN is harder to acquire after MEM-first training, we conduct further neuron-level analyses in three focal model–dataset settings.

Controlled continuations show that MEM-first models retain the ability to learn GEN: when subsequently optimized for GEN alone, the same MEM-first endpoints reach over 80% GEN in three focal cases. The difficulty instead appears when GEN must be acquired while MEM remains jointly optimized. This result motivates examining how functional organization of GEN and MEM emerges during joint training.

We identify a fixed GEN-associated neuron family and track its functional relevance as MEM is acquired. Ablating this family substantially reduces MEM after MEM is established under both training histories. Under GenMix, activation anchoring further shows that this functional dependence can emerge even when measured activation drift in the family is strongly constrained.

Together, these analyses show that early objective prioritization affects subsequent learning and reveal cross-capability functional dependence that emerges as GEN and MEM are learned together.

Our key contributions are summarized as follows:

1. **We identify a systematic asymmetry in early objective prioritization.** Across 54 model–dataset settings, GEN-first training produces higher subsequent GEN than MEM-first training, despite identical subsequent joint objectives.
2. **We show that the MEM-first disadvantage is conditional on joint optimization.** MEM-first models can still acquire strong GEN when optimized for GEN alone, but acquire substantially less GEN when MEM remains jointly optimized.
3. **We provide functional evidence of interaction between GEN and MEM during joint training.** MEM becomes functionally dependent on a fixed GEN-associated neuron family under both training histories; under GenMix, this dependence can emerge even when activation drift in that family is strongly constrained.

2 Related work

2.1 Training order, curriculum, and learning history

Curriculum learning establishes that data presentation order can affect optimization and generalization [2, 14, 35], while early-training studies show that what happens early in training can have lasting effects on later representations and performance [1, 25]. Recent LLM studies have extended these findings by investigating the interplay between early pretraining dynamics and factors such as model scale [9], data complexity [28], and learning-rate schedules [19, 5]. In particular, an analysis of pretraining dynamics finds that curricula mainly alter optimization within shared latent training phases rather than creating entirely new trajectories [9].

Our work studies a different question. Prior curriculum methods mainly change the order or selection of training examples, such as presenting easier examples before harder ones or prioritizing higher-quality data. We instead change what the model is trained to learn first: generalization (GEN) or memorization (MEM). We compare $\mathrm{GEN} \rightarrow (\mathrm{GEN} + \mathrm{MEM})$ with $\mathrm{MEM} \rightarrow (\mathrm{GEN} + \mathrm{MEM})$ to test whether early objective prioritization still affects learning after both models transition to the same joint objective.

2.2 Memorization and generalization

Neural networks can fit arbitrary labels while also learning generalizable structure [36], and memorization is widespread in language-model training [31, 4]. Grokking shows that memorization and generalization can emerge on different timescales [26], while subsequent mechanistic studies show that memorizing and generalizing solutions can develop through different internal dynamics [21, 32, 22].

Our setting directly controls which behavior is learned first rather than observing MEM and GEN emerge naturally during training, we train either GEN or MEM first, then train both models on the same joint GEN+MEM objective. This allows us to test whether early GEN or MEM learning still affects subsequent learning once the training objective becomes the same.

2.3 Sequential and multi-objective learning

Prior learning influences how neural networks acquire subsequent tasks or objectives through effects such as positive transfer and catastrophic interference [20, 17, 15, 8]. Related work in continual and multi-task learning has developed techniques to mitigate interference between objectives, including parameter protection and gradient-based approaches [17, 10].

Unlike continual learning, we study how prior learning affects later joint optimization by keeping both GEN and MEM active during the Phase-2, rather than forgetting after an objective is removed.

2.4 Mechanistic interpretability of learning dynamics

Prior work has identified functional structure within dense networks through sparse subnetworks and neuron-level analyses [11, 13], and used causal interventions to test their functional roles for specific behaviors [33, 7]. Our analysis builds on *Low-Complexity Probing via Finding Subnetworks* [3], which identifies functional subnetworks without fitting an expressive external probe. Related work has also examined how mechanisms form and evolve during training [23, 21]. We extend this perspective to study how GEN and MEM interact across learning histories. We identify a GEN-associated neuron family at a GEN-only reference endpoint, keep it fixed during mixed training, and examine whether it also becomes functionally relevant to MEM. Activation anchoring further tests whether this relevance can emerge without large changes in the family's activation state.

3 Experimental Setup

3.1 Tasks and Behavioral Objectives

We study two types of learning: generalization (GEN) and memorization (MEM). GEN learns the underlying task rule and applies it to unseen examples, and A_{GEN} is evaluated on held-out prompts disjoint from all training prompts. MEM learns fixed prompt–target associations that cannot be derived from the task rule through pairing training prompts with fixed, deliberately incorrect targets, and A_{MEM} measures recovery of these assigned targets.

We evaluate six model configurations across nine tasks, yielding 54 model–task settings. The models span scratch training, partial fine-tuning, and LoRA adaptation; the tasks are Addition, Three Operations, Color, Logical Deduction, Tracking Shuffled Objects, Object Counting, CLUTRR, GSM8K, and SVAMP-derived word problems. Task splits, MEM assignments, and evaluation subsets are shared across models. Dataset and model details are provided in Appendices A and B.

3.2 Training Regimes and Evaluation

All regimes use the same $2K$-step training budget and effective batch size. GenMix trains only on GEN for the first K steps and MemMix only on MEM, after which both switch to the same joint GEN+MEM training procedure for another K steps, with GEN and MEM sampled equally. PureMix instead trains jointly on GEN and MEM for all $2K$ steps. Thus, GenMix and MemMix differ only in which objective is prioritized during the first K steps; their subsequent joint training is identical. We report held-out GEN accuracy (A_{GEN}) and MEM accuracy (A_{MEM}) separately. Figure 1 summarizes the training regimes, with setting-specific training and evaluation details provided in Appendices B.2 and B.3.

3.3 Functional Neuron Family Identification

We conduct mechanistic case studies on MiniGPT–Addition, Qwen-0.8B–Addition, and Llama-3.2-3B-LoRA–Logical Deduction. For each case, we identify neuron families at a GEN-only reference endpoint using low-complexity subnetwork probing [3] with hard-concrete gates [18]. Model

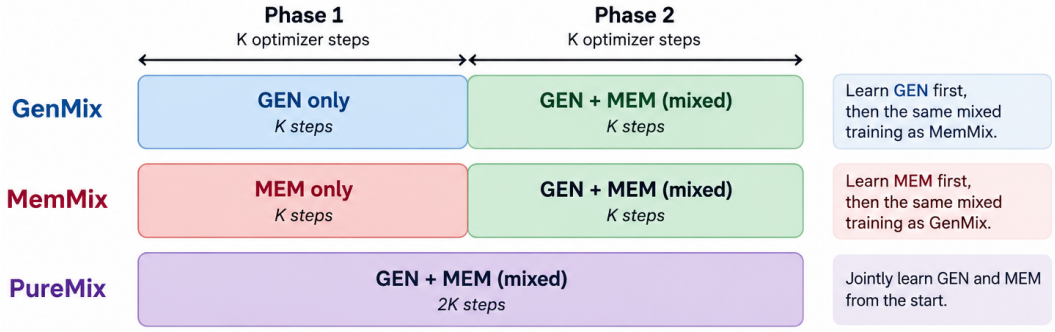


Figure 1: Training schedules. GenMix and MemMix train GEN or MEM for K steps before joint GEN+MEM training for another K steps. PureMix trains both objectives jointly for all $2K$ steps.

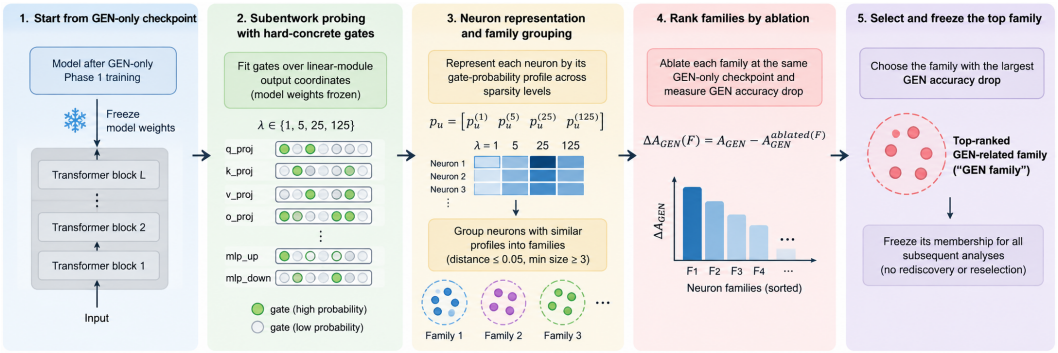


Figure 2: GEN-family identification and fixed selection. The top-ranked family at the GEN-only checkpoint is fixed for subsequent analyses.

weights are frozen during probing, and gates are fitted at $\lambda \in \{1, 5, 25, 125\}$. Neurons with similar gate profiles across these probes are then grouped into families.

We rank families by their GEN ablation effect on held-out GEN accuracy at the GEN-only endpoint,

$$\Delta A_{\mathrm{GEN}}(F) = A_{\mathrm{GEN}} - A_{\mathrm{GEN}}^{\mathrm{ablated}(F)}, \tag{1}$$

and select the family with the largest GEN accuracy drop as the *GEN-associated family*. Its membership is then kept fixed throughout the temporal-ablation and activation-anchoring analyses. The term "GEN-associated" refers only to how the family is selected and does not imply that it is unique to or exclusively responsible for GEN.

Full probing and family-construction details are provided in Appendix C.

3.4 Controlled Continuations and Interventions

We further perform three follow-up analyses based on the mechanistic cases.

First, controlled continuations vary the relative GEN and MEM loss weight from the Phase-1 endpoints to compare GEN learning with and without continued MEM optimization during Phase-2 mix training.

Second, we ablate the same fixed GEN-associated neuron family at successive Phase-2 checkpoints under GenMix and MemMix to track how its functional relevance to MEM evolves across learning histories.

Third, under GenMix Phase-2 mix training, we constrain activation drift in the same family while learning MEM, and then ablate the family after training to measure its functional relevance under the constraint.

Full methodological details are provided in Appendices D and E.

Table 1: Pairwise comparisons of final held-out GEN accuracy. Entries report the number (percentage) of model–dataset settings in which GenMix achieves higher GEN accuracy than the comparison regime.

Comparison	GenMix higher GEN
vs. MemMix	**40/54 (74.1%)**
vs. PureMix	36/54 (66.7%)

4 Results: Early objective prioritization Shapes Subsequent Learning

4.1 Early Objective Prioritization Shapes Subsequent Generalization

We evaluate the effect of early objective prioritization across 54 model–dataset settings spanning six models and nine tasks. Figure 3 compares final held-out GEN accuracy between GenMix and MemMix across all settings. GenMix achieves higher final GEN than MemMix in 40 of 54 settings (74.1%), compared with 12 MemMix wins and two ties. Mean final GEN is 44.49% for GenMix and 32.47% for MemMix, a 12.02 percentage-point advantage (Table 1 and appendix Table B.4). The win rate shows that this difference is not driven by only a few model–dataset pairs, although its magnitude varies across settings.

PureMix provides a joint-from-start reference to demonstrate when GEN and MEM are optimized together from the beginning, models can achieve performance on both. Overall, GEN-first training leads to stronger subsequent generalization than MEM-first training in nearly three quarters of the evaluated settings, even though both eventually use the same joint objective.

Multi-seed replication. We further run multi-seed replications on the same three mechanistic case studies with three seeds per case. GenMix achieves higher final GEN than MemMix in all nine paired runs. Mean GEN is 0.552 ± 0.083 versus 0.073 ± 0.025 for MiniGPT–Addition, 0.603 ± 0.018 versus 0.375 ± 0.038 for Qwen-0.8B–Addition, and 0.903 ± 0.006 versus 0.295 ± 0.184 for Llama-3.2-3B-LoRA–Logical Deduction (GenMix versus MemMix; mean $\pm$ s.d.). Final MEM remains high under both histories. Full per-seed results are reported in Appendix Table B.6. These multi-seed results confirm that the GenMix–MemMix difference is not limited to a single training run in the three mechanistic case studies.

This broad asymmetry raises a key question: why is GEN harder to acquire after MEM-first training than after GEN-first training? Does MEM-first training reduce the model's ability to learn GEN, or does the difficulty arise specifically when GEN and MEM are jointly optimized?

4.2 MEM-first training restricts joint GEN acquisition, not subsequent GEN learnability

To distinguish these two possibilities, we continue training from the GEN-only and MEM-only endpoints while varying the relative weight of the MEM objective. This allows us to compare GEN acquisition when GEN is optimized alone with when it is learned alongside MEM.

Starting from the final GEN-only or MEM-only Phase-1 checkpoint, we replace the standard Mix-stage objective with

$$\mathcal{L}_\alpha = (1-\alpha)\mathcal{L}_{\text{GEN}} + \alpha\mathcal{L}_{\text{MEM}}, \qquad \alpha \in \{0, 0.1, 0.25, 0.5, 0.75\}.$$

Each update uses equal numbers of GEN and MEM examples, so α controls their relative loss weights rather than their sampling frequency: $\alpha = 0$ corresponds to GEN-only optimization, $\alpha = 0.5$ assigns equal weight to both losses, and $\alpha = 0.75$ assigns three times as much weight to MEM as to GEN. Each branch independently starts from its corresponding Phase-1 checkpoint and is trained for the full Mix-stage budget. We report GEN and MEM accuracy at the final continuation step.

At $\alpha = 0$, removing the MEM objective allows substantial GEN acquisition from the MEM-first endpoints, while previously learned MEM largely disappears. The lower GEN under mixed training therefore cannot be explained simply by GEN becoming unlearnable after MEM-first training.

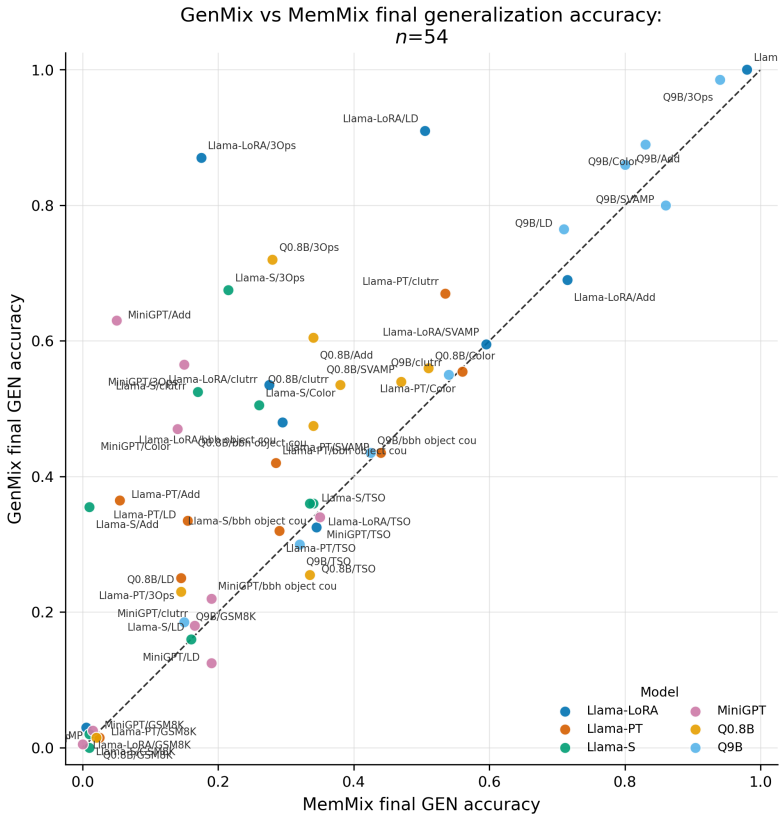


Figure 3: Final GEN accuracy for GenMix versus MemMix across 54 model–dataset pairs. Points above the diagonal favor GenMix. Full results are reported in Appendix B.4.

Once the MEM objective is active, however, the two histories separate substantially. Even when MEM performance is matched or nearly matched, GEN remains consistently higher after GEN-first training: this occurs at $\alpha = 0.5$ in MiniGPT–Addition, at every nonzero α in Qwen-0.8B–Addition, and across all nonzero α in Llama-3.2-3B-LoRA–Logical Deduction.

Together with the strong GEN recovery at $\alpha = 0$, these results distinguish GEN learnability from joint GEN acquisition. MEM-first checkpoints retain the ability to learn GEN when MEM is no longer optimized, but are substantially less effective at acquiring GEN while continuing to optimize and maintain MEM. Thus, within the tested Mix-stage conditions, the MEM-first disadvantage specifically concerns learning additional GEN under continued MEM optimization, rather than a general loss of subsequent GEN learnability.

Table 2: Final GEN/MEM accuracy under different objective weights after Mix-stage. Each cell reports GEN / MEM accuracy (%). α is the relative weight of the MEM loss in $\mathcal{L}_\alpha = (1-\alpha)\mathcal{L}_{\mathrm{GEN}} + \alpha\mathcal{L}_{\mathrm{MEM}}$; GEN and MEM sampling remains balanced. All results are measured at the final step of the corresponding Mix-stage continuation.

Setting	α	MEM-first	GEN-first
MiniGPT–Addition	0.00	100.0 / 0.0	100.0 / 0.0
	0.10	26.0 / 96.5	71.0 / 69.0
	0.25	12.5 / 98.0	57.5 / 93.5
	0.50	4.0 / 96.0	55.0 / 96.0
	0.75	3.0 / 99.0	49.5 / 98.5
Qwen-0.8B–Addition	0.00	96.0 / 0.0	100.0 / 0.0
	0.10	52.0 / 100.0	70.0 / 100.0
	0.25	43.0 / 100.0	61.0 / 100.0
	0.50	41.0 / 100.0	63.5 / 100.0
	0.75	37.0 / 100.0	57.5 / 100.0
Llama-3.2-3B-LoRA–Logical Deduction	0.00	80.5 / 1.5	92.0 / 0.5
	0.10	43.5 / 97.5	90.5 / 96.0
	0.25	37.0 / 99.0	90.5 / 97.0
	0.50	40.0 / 99.0	91.0 / 97.5
	0.75	38.5 / 99.5	91.0 / 97.0

4.3 Cross-Capability Functional Reliance on an Established GEN-Related Neuron Family

The controlled continuations in Section 4.2 show that MEM-first models retain the ability to learn GEN, but GEN acquisition is substantially reduced when MEM remains jointly optimized. This motivates examining whether GEN and MEM remain functionally separate during joint training. We therefore investigate whether MEM becomes functionally dependent on the fixed neuron family identified through GEN.

For each case, we use the top-ranked GEN-associated neuron family identified at the GEN-only reference endpoint (Appendix C). During the mixed stage of GenMix and MemMix, we measure MEM accuracy on the fixed confirmation set before and after ablating this family at successive checkpoints. Family membership remains fixed and is never selected or reranked using MEM behavior.

Figure 4 shows that once MEM is established, ablating the fixed GEN-associated family substantially reduces MEM accuracy under both histories. At the final checkpoint, MEM accuracy falls from 95.5% to 15.5% (GenMix) and 99.0% to 35.0% (MemMix) for MiniGPT; from 100% to 0% under both histories for Qwen; and from 99.5% to 22.0% and 99.5% to 4.5%, respectively, for Llama.

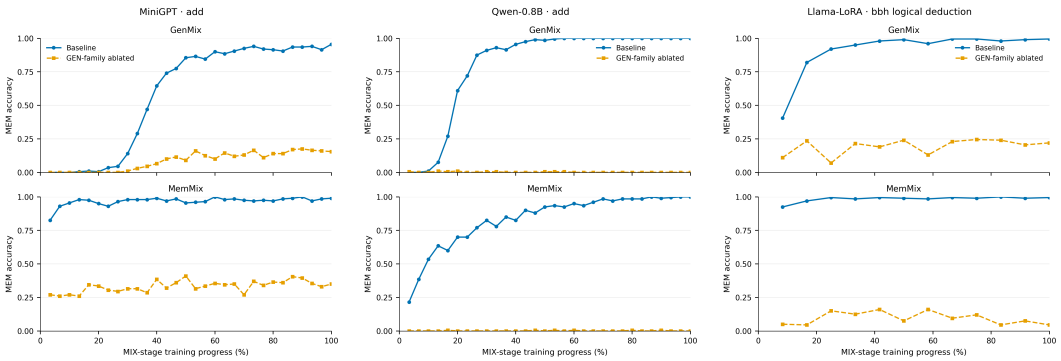
Thus, a family selected through GEN is also functionally relevant to MEM after MEM is acquired, revealing cross-capability functional relevance across distinct learning histories.

4.4 New Functional Dependence Can Emerge Without Large Activation Changes

Section 4.3 shows that a fixed GEN-associated neuron family can become functionally relevant to later MEM. We next examine whether this functional relevance requires large changes in the family's activation state. Starting from the GEN-only endpoint, we constrain activation drift during mixed GEN+MEM training using a frozen reference model and an activation-anchoring penalty:

$$\mathcal{L}_{\mathrm{anchor}}(B) = \mathcal{L}_{\mathrm{Mix}}(B) + \lambda_{\mathrm{anchor}} D_F(B), \tag{2}$$

where $\mathcal{L}\mathrm{Mix}(B)$ is the mean cross-entropy over supervised target tokens in mixed batch B, $D_F(B)$ is the normalized squared activation drift of the fixed GEN-associated family F from its frozen GEN-only reference, and $\lambda\mathrm{anchor}$ controls the penalty strength. Thus, $\lambda_{\mathrm{anchor}} D_F(B)$ penalizes activation drift without freezing parameters or replacing activations.



(a) MiniGPT–Addition (b) Qwen-0.8B–Addition (c) Llama-3.2-3B-LoRA–Logical Deduction

Figure 4: GEN-family ablation reduces MEM performance under both training histories. MEM accuracy with and without ablation of the fixed GEN-associated neuron family during GenMix (top) and MemMix (bottom), evaluated on the same fixed MEM confirmation set.

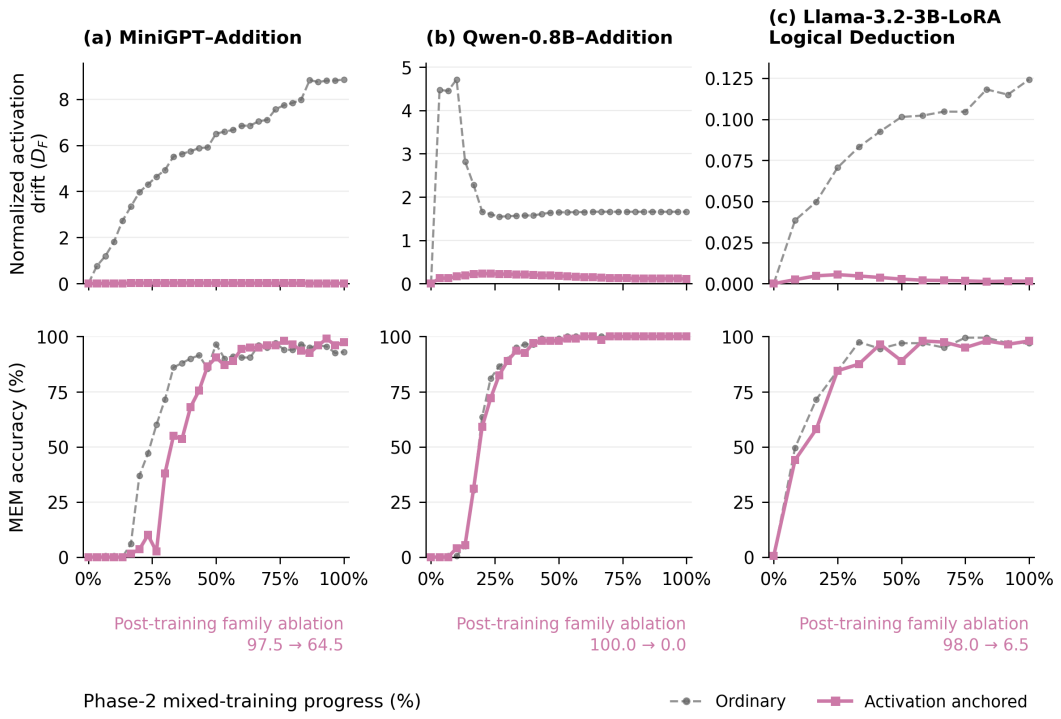


Figure 5: Activation anchoring suppresses GEN-family drift while MEM is acquired. Normalized activation drift of the fixed GEN-associated family (top) and MEM accuracy (bottom) during mixed training, with ordinary and activation-anchored continuations. Post-training annotations report the change in MEM accuracy after ablating the anchored family.

Under GenMix, activation anchoring reduces measured GEN-family drift by more than 93% in all three cases while each model still acquires high MEM accuracy (Figure 5). Thus, large measured activation changes in the fixed GEN-associated family are not required for MEM acquisition during GenMix. Importantly, the anchored family remains functionally relevant to MEM: ablating it after training substantially reduces MEM accuracy in all three cases. Together, these results show that, under GenMix, MEM can become functionally dependent on the fixed GEN-associated family even when its measured activation drift is strongly constrained.

5 DISCUSSION

Our results show that early capability prioritization can remain consequential after training transitions to a shared objective. Across models and tasks, prioritizing GEN first tends to produce stronger subsequent GEN while still allowing substantial MEM acquisition. Together with strong joint-from-start performance, this suggests that the key issue is not whether GEN and MEM can coexist, but how prior learning shapes their later integration.

Controlled continuations sharpen this distinction. MEM-first checkpoints retain substantial GEN learnability when optimized for GEN alone, but acquire less GEN when MEM remains jointly optimized. Early learning history therefore affects capability integration rather than simply determining whether a capability remains individually learnable.

Our neuron-level analyses provide a complementary functional view. Once MEM is established, ablating the same fixed GEN-associated family substantially impairs MEM under both GenMix and MemMix, showing cross-capability functional relevance. Activation anchoring further shows that this relevance can arise despite strongly constrained activation drift in the identified family. Thus, later capabilities can become functionally dependent on previously identified components without large changes in their measured activation state.

Our controlled GEN/MEM setting leaves open how these effects extend to richer capability combinations, larger-scale training, and longer horizons. The functional analysis is limited to one top-ranked GEN-associated family in three representative cases; its ablation establishes functional relevance, but neither selective localization nor a complete explanation of the GenMix–MemMix asymmetry. Future work could examine interactions across multiple representations and test whether similar patterns generalize across models, tasks, and capability combinations.

Finally, activation anchoring constrains measured activation drift rather than all changes in the surrounding computation. Functional dependence may instead arise through downstream readout, interactions with other representations, or changes elsewhere in the network. Identifying these mechanisms—particularly those underlying the difficulty of acquiring GEN while preserving MEM after MEM-first training—is an important direction for future work.

6 CONCLUSION

We show that what a model learns first can continue to shape what it learns later, even after training transitions to the same joint objective. MEM-first training does not eliminate GEN learnability, but GEN acquisition is substantially reduced when MEM remains jointly optimized. Our functional analyses further show that GEN and MEM can become dependent on the same identified neuron family during joint training. Together, these results establish early objective prioritization as an important factor in subsequent capability development. They motivate studying when different capabilities should be prioritized, and how these early choices affect later learning at larger scales and in more complex training settings.

AI USE STATEMENT

In this work, we used generative AI tools for hypothesis refinement and debugging of experimental code. Additionally, we used generative AI tools for writing and editing, literature discovery and summarization, and drafting and refining portions of the manuscript. We reviewed all AI-assisted work: research suggestions were independently evaluated by the authors, literature and citations were verified against the original sources, and AI-assisted code was reviewed and tested for correctness before use in experiments. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we provide the following resources:

- Code and Data Availability: The complete codebase—including training and evaluation scripts, configuration files, and the curated dataset splits—will be publicly released on GitHub upon acceptance.
- Implementation Details: Comprehensive descriptions of all model architectures, baseline setups, hyperparameter selections, and training procedures are documented in Appendix B.
- Datasets and Tasks: Detailed specifications of the nine tasks, synthetic generation rules, data preprocessing steps, and evaluation subsets are fully detailed in Appendix A.

REFERENCES

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkeStsCcKQ.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380.

[3] Steven Cao, Victor Sanh, and Alexander M. Rush. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 960–966. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.74.

[4] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

[5] Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for LLM reasoning. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2505.14970. Accepted for publication.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168.

[7] Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

[8] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7): 3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446.

[9] Mohamed Elgaar and Hadi Amiri. Curriculum learning for LLM pretraining: An analysis of learning dynamics. *arXiv preprint arXiv:2601.21698*, 2026. doi: 10.48550/arXiv.2601.21698.

[10] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 2020. URL https://proceedings.mlr.press/v108/farajtabar20a.html.

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

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

[13] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023. doi: 10.48550/arXiv.2305.01610.

[14] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2535–2544. PMLR, 2019. URL https://proceedings.mlr.press/v97/hacohen19a.html.

[15] Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2020. doi: 10.1016/j.tics.2020.09.004.

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

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017. doi: 10.1073/pnas.1611835114.

[18] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*, 2018.

[19] Kairong Luo, Zhenbo Sun, Haodong Wen, Xinyu Shi, Jiarui Cui, Chenyi Dang, Kaifeng Lyu, and Wenguang Chen. How learning rate decay wastes your best data in curriculum-based LLM pretraining. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T5wkZJqzkz.

[20] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower (ed.), *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8.

[21] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2301.05217.

[22] Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning. In *International Conference on Learning Representations*, pp. 46189–46209, 2025. URL https://arxiv.org/abs/2412.00104.

[23] 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://arxiv.org/abs/2209.11895. arXiv:2209.11895.

[24] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021. doi: 10.18653/v1/2021.naacl-main.168.

[25] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20596–20607, 2021. URL https://papers.neurips.cc/paper/2021/hash/ac56f8fe9eea3e4a365f29f0f1957c55-Abstract.html.

[26] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[27] Qwen Team. Qwen3.5: Towards native multimodal agents, 2026.

[28] Matthew Theodore Roque and Dan John Velasco. Beyond repetition: Text simplification and curriculum learning for data-constrained pretraining. In *Proceedings of the First BabyLM Workshop*, pp. 246–255. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.babylm-main.19.

[29] Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and R. Devon Hjelm. CLUTRR: A benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4506–4515, 2019. doi: 10.18653/v1/D19-1459.

[30] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. doi: 10.48550/arXiv.2206.04615.

[31] Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/fa0509f4dab6807e2cb465715bf2d249-Abstract-Conference.html.

[32] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023. doi: 10.48550/arXiv.2309.02390.

[33] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

[34] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015. doi: 10.48550/arXiv.1502.05698.

[35] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tW4QEInpni.

[36] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

A Dataset construction and objective definition

A.1 Dataset construction and splits

We evaluate nine tasks spanning algorithmic computation, symbolic and relational reasoning, and mathematical word problems. For each task, we construct separate GEN-training and MEM-training sets and a held-out GEN-test set. GEN- and MEM-training prompts are mutually disjoint, and GEN-test prompts are disjoint from both training sets. The same persisted datasets and target assignments are reused across model configurations and training histories.

Table A.1: Dataset split sizes.

Dataset	GEN train	MEM train / MEM test pool	GEN test pool
Addition	20,000	20,000	2,000
Three Operations	20,000	20,000	2,000
Color	5,000	5,000	2,000
Logical Deduction	2,000	2,000	2,000
Tracking Shuffled Objects	450	450	450
Object Counting	2,000	2,000	2,000
GSM8K	3,736	3,736	1,319
SVAMP	684	684	684
CLUTRR	1,500	1,500	1,500

Dataset construction. **Addition** uses disjoint three-digit operand pairs and tests whether models learn the addition rule on unseen inputs.

Three Operations extends this construction to addition, nonnegative subtraction, and floor integer division, with disjoint operands across splits.

Color is adapted from the induction task in the bAbI dataset [34]. It tests relational generalization through actor–animal and actor–color relations, with held-out actor triples used for GEN evaluation.

Logical Deduction and **Tracking Shuffled Objects** follow the formats of the corresponding BIG-Bench Hard tasks [30], but use newly generated examples rather than the unmodified benchmark instances. Logical Deduction contains object-ordering problems, while Tracking Shuffled Objects tracks object ownership after a sequence of swaps.

Object Counting is likewise a synthetic task following the corresponding BIG-Bench Hard format [30], with held-out prompts generated from the same construction distribution.

GSM8K [6] uses disjoint subsets of the official training set for GEN and MEM training and the official test set for GEN evaluation; answers are evaluated numerically.

SVAMP [24] is a controlled addition task constructed from usable SVAMP-derived templates.

CLUTRR [29] tests two-hop relational generalization using examples constructed from the published training and validation splits. Task splits, MEM assignments, and evaluation subsets are shared across models.

A.2 GEN and MEM objectives

GEN examples retain the rule-consistent target y_i defined by each task. GEN performance is evaluated on held-out prompts and therefore measures whether the learned task rule transfers beyond the training examples.

MEM examples preserve the corresponding task's input format but replace the correct target y_i with a deliberately incorrect target $\tilde{y}_i$, such that

$$\mathcal{D}_{\mathrm{MEM}} = \{(x_i, \tilde{y}_i)\}_{i=1}^{N}, \qquad \tilde{y}_i \neq y_i.$$

Because these assigned targets cannot be obtained by applying the underlying task rule, MEM performance measures recovery of arbitrary prompt–target associations rather than correct task execution.

MEM accuracy is

$$A_{\mathrm{MEM}} = \frac{1}{N_{\mathrm{eval}}} \sum_{i=1}^{N_{\mathrm{eval}}} \mathbf{1}[\hat{y}_i = \tilde{y}_i], \tag{A.1}$$

where agreement is measured against the assigned incorrect target $\tilde{y}_i$ rather than the rule-consistent target y_i.

Table A.2: MEM-target construction policies.

Dataset	MEM-target construction
Addition	Random integer in $[200, 1998]$, excluding the correct sum.
Three Operations	Random valid task-range value, excluding the correct answer.
Color	Incorrect color sampled from the allowed color set.
Logical Deduction	Incorrect valid answer option.
Tracking Shuffled Objects	Incorrect valid answer option.
Object Counting	Seeded permutation of correct answers within each category–item-count stratum, constrained so that every assigned target differs from the corresponding correct answer.
CLUTRR	Incorrect relation sampled from the other 17 relation labels.
GSM8K	Deterministic permutation of answers across prompts, constrained so that each assigned MEM target differs from the prompt's correct answer.
SVAMP	Answer sampled from the available answer pool, excluding the correct answer.

Unlike GEN evaluation, MEM evaluation intentionally reuses the MEM-training prompts together with their fixed assigned targets. This is necessary because the MEM objective is defined as retention of prompt-specific arbitrary associations; there is no rule-consistent mapping from which the assigned incorrect targets could be inferred for unseen prompts.

A.3 Example GEN and MEM Instances

We provide one MEM and one GEN example from each dataset to illustrate the two learning objectives. GEN training and test examples follow the same format and use correct task targets, so we show only one GEN example per dataset. MEM examples instead use deliberately assigned incorrect targets that cannot be derived from the task rule. For each MEM example, we also report the correct answer for comparison. Long prompts are shortened for display, while all targets are shown as used in training.

Table A.3: Example GEN and MEM instances across the nine datasets. GEN maps each prompt to its correct task target. MEM instead maps the prompt to an assigned incorrect target; the correct task answer is shown in parentheses.

Dataset	GEN example	MEM example
Addition	`157+299=` → `_456#`	`572+855=` → `_730#` (correct: `1427#`)
Three Operations	`694-488=` → `___206#`	`480-384=` → `___870#` (correct: `____96#`)
Color	`Harrison is crocodile. Harrison has_color coral. ... What color is Abigail?` → `coral`	`Alice has_color cyan. Bob has_color aqua. ... What color is Charlie?` → `mustard` (correct: `aqua`)
BBH Logical Deduction	`There are 7 objects ... Which object is immediately to the left of the purple pen?` → `D`	`There are 5 objects ... Which object is in the fourth position from the left?` → `C` (correct: `A`)
BBH Tracking Shuffled Objects	`Cara: maroon tile. Gina: yellow spool. Priya: gray token. ... Who has the gray token?` → `C`	`Alice: amber book. Bob: blue cup. Cara: brass key. ... Who has the amber book?` → `A` (correct: `B`)
BBH Object Counting	`I have a peach, an orange, ... and two raspberries. How many fruits do I have?` → `8`	`I have an oven, four chairs, ... and a microwave. How many objects do I have?` → `9` (correct: `10`)
SVAMP	`Tiffany ... found 449 bags ... and 16 bags ... How many bags did she find after monday?` → `465`	`Paul ... gave 681 crayons ... and lost 158 ... How many crayons had been lost or given away?` → `1707` (correct: `839`)
GSM8K	`An eraser costs $2 and a pencil costs $3. How much do 6 erasers and 8 pencils cost?` → `36`	`Lee had $10 and his friend had $8 ... How much change should they have received in total?` → `160` (correct: `3`)
CLUTRR	`[Jason] ... his son [Christopher] ... How is Jason related to Christopher?` → `father`	`[James] and his brother [Dwight] ... [James]'s sister [Myrna] ... How is Myrna related to Dwight?` → `grandfather` (correct: `sister`)

Table B.1: Experimental coverage, model configurations, and trainable components. Each model configuration is evaluated on all nine tasks.

Model	Initialization	Architecture	Trainable parameters
MiniGPT	Scratch	Six Transformer blocks, hidden size 384, six attention heads	All parameters
Qwen-0.8B	Pretrained	Qwen3.5-0.8B-Base [27], 24 language blocks	Final six language blocks
Llama-3B	Pretrained	Llama-3.2-3B [12], 28 layers	Final seven layers
Llama-Scratch	Scratch	Custom Llama-style architecture: 12 layers, hidden size 768, intermediate size 3072, eight attention heads, and eight KV heads	All parameters
Llama-3B-LoRA	Pretrained	Llama-3.2-3B [12] with query/value LoRA [16]	LoRA adapters
Qwen-9B	Pretrained	Qwen3.5-9B-Base [27], 32 language blocks	Final eight language blocks

B MODELS, TRAINING, AND COMPLETE BEHAVIORAL RESULTS

B.1 MODEL CONFIGURATIONS

We evaluate six model configurations spanning training from scratch, partial fine-tuning of pretrained models, and parameter-efficient adaptation. Model configurations, initialization, and trainable parameters are summarized in Table B.1.

B.2 TRAINING HISTORIES AND BUDGETS

For every model–dataset pair, we compare three histories under a matched total budget of $2K$ optimizer updates:

$$\begin{aligned} \text{GenMix}: &\quad \text{GEN}_K \rightarrow (\text{GEN} + \text{MEM})_K, \\ \text{MemMix}: &\quad \text{MEM}_K \rightarrow (\text{GEN} + \text{MEM})_K, \\ \text{PureMix}: &\quad (\text{GEN} + \text{MEM})_{2K}. \end{aligned} \tag{B.1}$$

During mixed training, GEN and MEM examples are sampled with equal probability. Thus, the three histories use the same total optimization budget within each model–dataset setting while differing in their early objective history.

Table B.2: Phase length K in thousands of optimizer updates.

Dataset	MiniGPT	Qwen-0.8B	Llama pretrained	Llama scratch	Llama-LoRA	Qwen-9B
Addition	30	30	6	60	12	10
Three Operations	50	50	40	60	12	10
Color	40	10	4	21	13	2
Logical Deduction	10	10	9	20	12	3
Tracking Shuffled Objects	15	6	8	15	5	4
Object Counting	20	10	9	20	12	4
GSM8K	20	5	8	10	11	2
SVAMP	10	9	8	20	12	2
CLUTRR	20	10	9	20	12	4

All behavioral-training settings use AdamW. Table B.3 reports the learning rate, microbatch size, gradient accumulation, effective batch size, and weight decay.

Table B.3: Behavioral-training hyperparameters.

Model	Dataset	LR	Microbatch × accum.	Effective batch	WD
MiniGPT	Addition	5×10^{-4}	256×1	256	0
	Three Operations	1×10^{-4}	256×1	256	0
	Color	1×10^{-4}	64×2	128	.01
	Logical Deduction	5×10^{-5}	32×1	32	.01
	Object Counting	1×10^{-4}	8×8	64	.01
	Tracking	1×10^{-4}	64×1	64	.01
	GSM8K	1×10^{-4}	8×8	64	.01
	SVAMP	1×10^{-4}	16×4	64	.01
	CLUTRR	1×10^{-4}	8×8	64	.01
Qwen-0.8B	Addition	1×10^{-4}	64×1	64	.01
	Three Operations	1×10^{-4}	64×1	64	.01
	Color	5×10^{-5}	4×4	16	.01
	Logical Deduction	5×10^{-5}	4×2	8	.01
	Tracking	5×10^{-5}	4×2	8	.01
	Object Counting	5×10^{-5}	4×2	8	.01
	GSM8K	5×10^{-5}	1×8	8	.01
	SVAMP	5×10^{-5}	4×4	16	.01
	CLUTRR	5×10^{-5}	4×2	8	.01
Llama pretrained	Addition	5×10^{-5}	128×1	128	.01
	Three Operations	1×10^{-4}	128×1	128	.01
	Color	5×10^{-5}	4×4	16	.01
	Logical Deduction	2×10^{-5}	1×8	8	.01
	Tracking	5×10^{-5}	8×1	8	.01
	Object Counting	2×10^{-5}	1×8	8	.01
	GSM8K	5×10^{-5}	1×8	8	.01
	SVAMP	5×10^{-5}	1×8	8	.01
	CLUTRR	2×10^{-5}	1×8	8	.01
Llama scratch	Addition	1×10^{-3}	128×1	128	0
	Three Operations	1×10^{-4}	128×1	128	.01
	Color	6×10^{-4}	8×4	32	.01
	Logical Deduction	3×10^{-4}	4×8	32	.01
	Tracking	3×10^{-4}	4×4	16	.01
	Object Counting	3×10^{-4}	4×8	32	.01
	GSM8K	3×10^{-4}	1×16	16	.01
	SVAMP	3×10^{-4}	4×8	32	.01
	CLUTRR	3×10^{-4}	4×8	32	.01
Llama-LoRA	Addition	5×10^{-5}	4×4	16	.01
	Three Operations	5×10^{-5}	8×4	32	.01
	Color	5×10^{-5}	4×4	16	.01
	Logical Deduction	5×10^{-5}	4×4	16	.01
	Tracking	5×10^{-5}	2×4	8	.01
	Object Counting	5×10^{-5}	4×4	16	.01
	GSM8K	5×10^{-5}	2×4	8	.01
	SVAMP	5×10^{-5}	2×4	8	.01
	CLUTRR	5×10^{-5}	4×4	16	.01
Qwen-9B	Addition	1×10^{-5}	1×8	8	.01
	Three Operations	1×10^{-5}	1×8	8	.01
	Color	1×10^{-5}	1×8	8	.01
	Logical Deduction	1×10^{-5}	1×8	8	.01
	Tracking	1×10^{-5}	1×8	8	.01
	Object Counting	5×10^{-5}	4×4	16	.01
	GSM8K	1×10^{-5}	1×8	8	.01
	SVAMP	1×10^{-5}	1×8	8	.01

Continued on next page

Table B.3: Behavioral-training hyperparameters (continued).

Model	Dataset	LR	Microbatch × accum.	Effective batch	WD
	CLUTRR	5×10^{-5}	4×4	16	.01

B.3 Behavioral Evaluation

We evaluate GEN and MEM separately. Let $\mathcal{D}^{\text{eval}}_{\text{GEN}}$ denote the fixed set of 200 held-out GEN examples and $\mathcal{D}^{\text{eval}}_{\text{MEM}}$ the fixed set of 200 MEM-training examples used for evaluation. We define

$$A_{\text{GEN}} = \frac{1}{|\mathcal{D}^{\text{eval}}_{\text{GEN}}|} \sum_{(x,y)\in\mathcal{D}^{\text{eval}}_{\text{GEN}}} \mathbf{1}[\hat{y}(x) = y]\,, \tag{B.2}$$

where y is the correct task answer, and

$$A_{\text{MEM}} = \frac{1}{|\mathcal{D}^{\text{eval}}_{\text{MEM}}|} \sum_{(x,\tilde{y})\in\mathcal{D}^{\text{eval}}_{\text{MEM}}} \mathbf{1}[\hat{y}(x) = \tilde{y}]\,, \tag{B.3}$$

where $\tilde{y}$ is the deliberately assigned target used during MEM training. Thus, A_{GEN} measures generalization to unseen prompts, whereas A_{MEM} measures retention of arbitrary associations presented during training.

The same GEN and MEM evaluation examples are used across checkpoints and training regimes within each model–dataset setting.

B.4 Complete behavioral results

This section reports the complete behavioral results underlying the aggregate comparisons in Section 4.1. We evaluate 54 model–dataset configurations, obtained from six model settings across nine tasks, under each of the three training regimes (GenMix, MemMix, and PureMix), yielding 162 behavioral results in total. For each configuration and regime, we report both held-out GEN accuracy and MEM accuracy on the assigned prompt–target associations.

Table B.4 summarizes aggregated results across the full evaluation matrix. Mean GEN and Mean MEM are computed over the 54 model–dataset configurations for each training regime. Pairwise contrasts report the difference between regime-level means, while GEN W/L/T counts summarize, across the same 54 configurations, how often the first regime achieves higher, lower, or equal GEN accuracy relative to the second regime. Table B.5 provides the corresponding configuration-level GEN/MEM accuracies from which these aggregate statistics are computed. Aggregate accuracies in Table B.4 are reported on the $[0, 1]$ scale, whereas the pair-level values in Table B.5 are reported as percentages for readability.

Table B.4: Full-matrix means and pairwise contrasts across 54 model–dataset configurations; W/L/T denotes GEN wins/losses/ties for the first regime relative to the second. Differences are absolute accuracy points.

Regime	Mean GEN	Mean MEM	
GenMix	.444907	.779722	
MemMix	.324722	.830093	
PureMix	.428426	.802315	
Contrast	**ΔGEN**	**ΔMEM**	**GEN W/L/T**
GenMix − MemMix	+.120185	−.050370	40/12/2
GenMix − PureMix	+.016481	−.022593	36/14/4
MemMix − PureMix	−.103704	+.027778	13/39/2

Table B.5: Complete pair-level behavioral results. Each entry reports GEN/MEM accuracy in percent.

Model	Dataset	GenMix G/M	MemMix G/M	PureMix G/M
MiniGPT	Addition	63.0/95.5	5.0/97.5	56.0/90.5
	Three Operations	56.5/98.5	15.0/98.5	58.5/100
	Color	47.0/100	14.0/100	36.5/100
	Logical Deduction	12.5/100	19.0/100	12.0/100
	Tracking	34.0/100	35.0/100	38.0/100
	Object Counting	22.0/100	19.0/100	17.0/100
	GSM8K	2.5/98.5	1.5/99.0	2.0/99.0
	SVAMP	0.5/100	0/100	0/100
	CLUTRR	18.0/100	16.5/100	15.5/100
Qwen-0.8B	Addition	60.5/100	34.0/100	59.0/100
	Three Operations	72.0/100	28.0/100	71.5/100
	Color	56.0/100	51.0/100	55.5/100
	Logical Deduction	23.0/100	14.5/100	18.0/100
	Tracking	25.5/42.5	33.5/99.0	32.5/56.0
	Object Counting	47.5/100	34.0/100	36.0/100
	GSM8K	1.5/95.5	2.0/100	1.0/100
	SVAMP	54.0/100	47.0/100	63.0/100
	CLUTRR	53.5/100	38.0/100	49.5/100
Llama pretrained	Addition	36.5/98.0	5.5/99.0	23.0/100
	Three Operations	25.0/100	14.5/100	27.5/100
	Color	55.5/99.0	56.0/100	56.5/100
	Logical Deduction	33.5/100	15.5/100	28.5/100
	Tracking	32.0/99.5	29.0/100	38.5/100
	Object Counting	42.0/100	28.5/100	34.5/100
	GSM8K	1.5/100	2.5/100	2.0/100
	SVAMP	43.5/100	44.0/100	60.5/100
	CLUTRR	67.0/100	53.5/100	54.5/100
Llama scratch	Addition	35.5/0	1.0/0	40.5/0
	Three Operations	67.5/99.5	21.5/99.0	67.0/97.5
	Color	50.5/1.5	26.0/2.5	46.0/2.5
	Logical Deduction	16.0/22.5	16.0/22.5	15.5/28.5
	Tracking	36.0/33.0	33.5/33.0	33.5/33.0
	Object Counting	36.0/33.5	34.0/22.0	33.5/15.0
	GSM8K	2.0/50.0	1.0/72.5	2.0/21.0
	SVAMP	0/99.5	1.0/100	0/100
	CLUTRR	52.5/86.5	17.0/97.0	38.5/96.5
Llama-LoRA	Addition	69.0/0	71.5/1.0	65.0/2.0
	Three Operations	87.0/4.0	17.5/31.5	71.5/22.5
	Color	100/92.5	98.0/98.5	100/99.0
	Logical Deduction	91.0/98.0	50.5/98.5	87.5/99.0
	Tracking	32.5/98.0	34.5/100	28.0/99.0
	Object Counting	48.0/99.0	29.5/99.5	42.5/99.5
	GSM8K	3.0/92.0	0.5/97.5	1.5/97.5
	SVAMP	59.5/100	59.5/98.5	63.5/98.5
	CLUTRR	53.5/99.5	27.5/100	50.0/100
Qwen-9B	Addition	86.0/0	80.0/1.5	83.0/1.0
	Three Operations	98.5/0	94.0/2.0	98.5/0
	Color	89.0/1.5	83.0/21.0	87.0/5.0
	Logical Deduction	76.5/96.0	71.0/100	74.0/98.5
	Tracking	30.0/43.5	32.0/100	36.0/100
	Object Counting	43.5/82.0	42.5/99.5	48.5/96.5
	GSM8K	18.5/51.5	15.0/92.0	15.5/75.0
	SVAMP	80.0/100	86.0/100	84.5/100
	CLUTRR	55.0/100	54.0/100	53.5/100

Table B.6: Multi-seed replication for the three settings used in subsequent analyses. Pre-GEN and Pre-MEM denote accuracies at the end of the initial single-objective phase; Final GEN and Final MEM denote accuracies after the full training schedule. PureMix has no single-objective phase.

Setting	Regime	Seed	Pre-GEN	Pre-MEM	Final GEN	Final MEM
MiniGPT–Addition	GenMix	0	1.000	0.000	0.630	0.955
	MemMix	0	0.000	0.965	0.050	0.975
	PureMix	0	–	–	0.560	0.905
	GenMix	1	1.000	0.000	0.560	0.935
	MemMix	1	0.000	0.990	0.070	0.995
	PureMix	1	–	–	0.515	1.000
	GenMix	2	1.000	0.000	0.465	0.945
	MemMix	2	0.000	0.980	0.100	0.990
	PureMix	2	–	–	0.505	1.000
Qwen-0.8B–Addition	GenMix	250	0.995	0.000	0.605	1.000
	MemMix	250	0.000	0.560	0.340	1.000
	PureMix	250	–	–	0.590	1.000
	GenMix	251	1.000	0.000	0.620	1.000
	MemMix	251	0.000	1.000	0.370	1.000
	PureMix	251	–	–	0.590	1.000
	GenMix	252	1.000	0.000	0.585	1.000
	MemMix	252	0.000	1.000	0.415	1.000
	PureMix	252	–	–	0.600	1.000
Llama-LoRA–Logical Deduction	GenMix	2024	0.925	0.015	0.910	0.980
	MemMix	2024	0.100	0.990	0.505	0.985
	PureMix	2024	–	–	0.875	0.990
	GenMix	2025	0.930	0.030	0.900	0.990
	MemMix	2025	0.070	0.975	0.220	0.995
	PureMix	2025	–	–	0.905	1.000
	GenMix	2026	0.940	0.020	0.900	0.985
	MemMix	2026	0.090	1.000	0.160	0.995
	PureMix	2026	–	–	0.870	0.995

All aggregate means, pairwise differences, and GEN win/loss/tie counts reported in Table B.4 and Section 4.1 are computed directly from the 162 configuration-level results in Table B.5. Thus, the aggregate comparisons and the complete behavioral matrix correspond to the same underlying experimental results.

B.5 Multi-seed behavioral replication

We replicate all three training regimes across three independent seeds for the three model–dataset settings used in the subsequent controlled and functional analyses: MiniGPT–Addition, Qwen-0.8B–Addition, and Llama-3.2-3B-LoRA–Logical Deduction. Within each setting, the dataset, GEN/MEM splits, assigned MEM targets, training budget, hyperparameters, and evaluation examples are held fixed across seeds. Seeds vary the stochastic training trajectory, including model initialization for scratch models and training-time stochasticity. Table B.6 reports all seed-level results.

Across the nine paired GenMix–MemMix comparisons, GenMix achieves higher final GEN accuracy in every case. The per-seed GenMix–MemMix GEN differences are $+0.580$, $+0.490$, and $+0.365$ for MiniGPT–Addition; $+0.265$, $+0.250$, and $+0.170$ for Qwen-0.8B–Addition; and $+0.405$, $+0.680$, and $+0.740$ for Llama-3.2-3B-LoRA–Logical Deduction. Final MEM remains high for both histories, indicating that the replicated GEN differences do not arise from failure to retain MEM at the end of joint training.

Table B.7: Final GEN accuracy across three independent seeds in the focal settings. Values are mean ± s.d.

Setting	GenMix	MemMix	PureMix
MiniGPT–Addition	0.552 ± 0.083	0.073 ± 0.025	0.527 ± 0.029
Qwen-0.8B–Addition	0.603 ± 0.018	0.375 ± 0.038	0.593 ± 0.006
Llama-LoRA–Logical Deduction	0.903 ± 0.006	0.295 ± 0.184	0.883 ± 0.019

C IDENTIFICATION OF GEN-ASSOCIATED NEURON FAMILIES

We identify one fixed GEN-associated neuron family for each of the three mechanistic cases. The procedure defines an analyzed neuron-coordinate space, estimates coordinate importance across multiple sparsity levels, groups coordinates with similar importance profiles into families, and selects the top-ranked family by its causal effect on held-out GEN. The selected family is then held fixed in the subsequent temporal-ablation and activation-anchoring analyses.

C.1 MECHANISTIC CASES AND NEURON SPACES

Table C.1 summarizes the three model–task pairs.

Table C.1: Mechanistic cases and selected GEN-associated families. Coordinates denotes the size of the analyzed neuron space; GEN family denotes the number of coordinates in the selected family.

Case	GEN-only endpoint	Trainable parameters	Coordinates	GEN family	Fraction
MiniGPT–Addition	30k	Full model	20,736	2,351	11.34%
Qwen-0.8B–Addition	30k	Final six language blocks	98,432	7,541	7.66%
Llama-LoRA–Logical Deduction	12k	Query/value LoRA	114,688	9,725	8.48%

We define a neuron operationally as one output coordinate of a linear module. For MiniGPT, we analyze linear-module outputs within the transformer blocks, including attention and feed-forward projections. For Qwen, we analyze linear-module outputs in the final six trainable language blocks. For Llama-LoRA, we analyze the output coordinates of `q_proj` and `v_proj` across all 28 language layers, where each output includes both the frozen base projection and its LoRA update. We do not separately define neurons over the internal LoRA low-rank dimensions. Embeddings,normalization layers, and the language-model head are excluded.

C.2 SUBNETWORK PROBING AND FAMILY CONSTRUCTION

At the GEN-only endpoint, model parameters are frozen and hard-concrete gates are optimized over the analyzed coordinates following the low-complexity subnetwork-probing framework. We fit four separately optimized probes with maximum sparsity coefficients

$$\lambda_{\max} \in \{1, 5, 25, 125\},$$

each initialized and run with seed 0.

Each probe is optimized for 1,500 steps using Adam with learning rate 0.2, $\beta = (0.9, 0.999)$, zero weight decay, and no gradient clipping. Probe batch sizes are 32, 4, and 1 for MiniGPT, Qwen, and Llama-LoRA, respectively, and model dropout is disabled. Training examples are sampled with replacement.

The probe objective is

$$\mathcal{L}_{\text{probe}} = \mathcal{L}_{\text{target-only CE}} + \lambda_t \frac{1}{N} \sum_{i=1}^{N} P(z_i > 0), \tag{C.1}$$

where N is the number of analyzed coordinates, z_i is the hard-concrete gate for coordinate i, and λ_t is the sparsity coefficient at optimization step t. The term $P(z_i > 0)$ denotes the hard-concrete gate probability for coordinate i. The sparsity coefficient is zero during the first quarter of optimization, increases linearly to $\lambda_{\max}$ during the middle half, and remains at $\lambda_{\max}$ for the final quarter. Hard-concrete gates use temperature $\beta_{\text{HC}} = 2$, $\gamma = -0.1$, and $\zeta = 1.1$, with gate logits initialized to zero.

Each coordinate u is represented by its gate probabilities across the four sparsity levels,

$$p_u = [p_u^{(1)}, p_u^{(5)}, p_u^{(25)}, p_u^{(125)}]. \tag{C.2}$$

Coordinates with similar signatures are grouped using deterministic single-pass nearest-centroid clustering. For a coordinate signature p and a family centroid c, we use the mean absolute distance

$$d(p, c) = \frac{1}{4} \sum_{j=1}^{4} |p_j - c_j| \tag{C.3}$$

A coordinate joins the nearest family satisfying $d(p, c) \leq 0.05$; otherwise, a new family is created. Families containing fewer than three coordinates are discarded.

C.3 Family Ranking and Fixed Selection

At the GEN-only endpoint, each retained family F is ablated and ranked by its effect on held-out GEN accuracy:

$$\Delta A_{\mathrm{GEN}}(F) = A_{\mathrm{GEN}} - A_{\mathrm{GEN}}^{\mathrm{ablated}(F)}. \tag{C.4}$$

Families are ranked by decreasing ΔA_{GEN}, with ties resolved by choosing the smaller family ID. The highest-ranked family is selected as the GEN-associated neuron family, abbreviated as the *GEN family*.

This selection is performed once at the GEN-only endpoint. The selected family contains 2,351, 7,541, and 9,725 coordinates in MiniGPT, Qwen, and Llama-LoRA, respectively, and its membership is held fixed throughout the subsequent temporal-ablation and activation-anchoring analyses. It is never rediscovered or reranked using later checkpoints or intervention outcomes.

The term *GEN-associated* refers only to this selection criterion. It does not imply that the selected family is uniquely or exclusively involved in GEN.

C.4 Mechanistic Evaluation Sets

The subsequent learning-dynamics analyses use fixed confirmation sets of 200 GEN and 200 MEM examples for each mechanistic case. The GEN confirmation examples are held out from training, while the MEM confirmation examples evaluate the assigned MEM targets on MEM-training prompts. These confirmation sets are kept fixed across checkpoints and learning histories so that temporal changes in accuracy and ablation effects are evaluated on the same examples.

The confirmation examples are separate from the examples used to calibrate the mechanistic analyses and exclude the original evaluation IDs. GEN prompt disjointness is additionally checked when constructing the confirmation set.

D CONTROLLED CONTINUATIONS AND FUNCTIONAL INTERVENTIONS

D.1 CONFIRMATION EVALUATION

The main mechanistic intervention results use fixed confirmation sets containing 200 held-out GEN examples and 200 MEM-training examples. The GEN confirmation examples are disjoint from GEN- and MEM-training prompts, while MEM confirmation evaluates recall of the deliberately assigned training associations.

These confirmation sets are distinct from the calibration examples used for activation anchoring below. Family membership is fixed before any continuation or anchoring outcome is observed.

D.2 CONTROLLED-CONTINUATION SETUP

The controlled continuations in Section 4.2 start independently from the stored GEN-only or MEM-only Phase-1 endpoints. Each branch runs for the full Phase-2 continuation budget.

Table D.1: Controlled-continuation settings.

Setting	MiniGPT–Addition	Qwen-0.8B–Addition	Llama-LoRA–Logical Deduction
Phase-1 checkpoint seed	0	250	2024
Phase-1 endpoint	30k	30k	12k
Continuation updates	30k	30k	12k
Continuation seed	0	0	0
Microbatch $\times$ accumulation	256×1	64×1	4×4
Effective batch size	256	64	16
Learning rate	5×10^{-4}	1×10^{-4}	5×10^{-5}
Weight decay	0	0.01	0.01
Trainable parameters	Full model	Final six blocks	Query/value LoRA

All branches use fresh AdamW optimizers with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and global gradient clipping at 1.0. Each optimizer update contains equal numbers of GEN and MEM examples, sampled uniformly with replacement from their respective training pools. This batch composition is held fixed across all values of α.

Because the original optimizer states are not reused, these experiments are controlled continuations from fixed Phase-1 model endpoints rather than exact replays of the original Phase-2 trajectories.

D.3 OBJECTIVE-WEIGHT CONTINUATIONS

To distinguish subsequent GEN learnability from GEN acquisition while MEM remains actively optimized, we vary the relative MEM loss weight during Phase-2 continuation:

$$\mathcal{L}_\alpha = (1-\alpha)\mathcal{L}_{\text{GEN}} + \alpha\mathcal{L}_{\text{MEM}}, \qquad \alpha \in \{0, 0.1, 0.25, 0.5, 0.75\}. \tag{D.1}$$

For each objective, the loss is the mean cross-entropy over valid target tokens across the accumulated microbatches. Because the GEN/MEM batch composition is fixed, α changes only their relative contribution to the optimization objective, not their example exposure.

At $\alpha = 0$, the continuation optimizes GEN alone regardless of the Phase-1 starting history. MEM examples are still evaluated in the forward pass but contribute no gradient. Thus, the MEM-first $\alpha = 0$ branch tests whether GEN remains subsequently learnable after MEM-first training, while the nonzero-α branches test GEN acquisition when MEM continues to be actively optimized.

The same objective-weight sweep is run independently from the MEM-first and GEN-first Phase-1 endpoints. Final GEN and MEM accuracies are measured after the full continuation budget and reported in Section 4.2.

E GEN-FAMILY ACTIVATION ANCHORING

E.1 ACTIVATION-ANCHOR INTERVENTION

Starting from the GEN-only endpoint, we retain a frozen copy of the model as an activation reference and continue mixed GEN+MEM training while penalizing activation drift in the fixed associated top-ranked GEN family F. For a sampled microbatch B, the anchored objective is

$$\mathcal{L}_{\text{anchor}}(B) = \mathcal{L}_{\text{Mix}}(B) + \lambda_{\text{anchor}} D_F(B), \tag{E.1}$$

where $\mathcal{L}_{\text{Mix}}(B)$ is the mean cross-entropy over all supervised target tokens in B, $D_F(B)$ is the normalized squared activation drift of F relative to the frozen GEN-only reference, and λ_{anchor} controls the strength of the drift penalty. Thus, the anchor constrains measured activation drift without replacing activations in the forward pass or introducing additional parameter freezing.

Mixed examples are sampled uniformly with replacement from the concatenated GEN and MEM training sets, without enforcing a fixed GEN/MEM ratio per microbatch. When gradient accumulation is used, microbatch objectives are averaged equally within each update; the accumulation factor is 1 for MiniGPT and Qwen and 4 for Llama-LoRA.

To define $D_F(B)$, let S_m denote the coordinates of F contained in module m, and let $\mathcal{M} = \{m : S_m \neq \emptyset\}$ denote the modules containing at least one coordinate of F. Let $a_{0,m,p,i}$ and $a_{\theta,m,p,i}$ denote the corresponding module-output activations of the frozen reference and current model at teacher-forced input position p and selected coordinate i. For each module, we compute a fixed reference scale

$$s_m = \max\left(\frac{1}{|V(C)|\,|S_m|} \sum_{p \in V(C)} \sum_{i \in S_m} a_{0,m,p,i}^2, 10^{-6} \right), \tag{E.2}$$

where C is the calibration set and $V(C)$ denotes the input positions used for activation measurement.

For microbatch B, normalized drift is

$$D_F(B) = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \frac{1}{s_m} \frac{1}{|V(B)|\,|S_m|} \sum_{p \in V(B)} \sum_{i \in S_m} \left(a_{\theta,m,p,i} - a_{0,m,p,i}\right)^2, \tag{E.3}$$

where $V(B)$ denotes the teacher-forced input positions used by the activation anchor, including prompt positions; padding positions are excluded for Llama-LoRA. The activation mask is independent of the supervised-label mask. Within each module, squared drift is averaged jointly over positions and selected coordinates, after which modules are averaged equally regardless of the number of selected coordinates they contain.

The anchor is a soft activation constraint. Reference activations are observed without being substituted into the current forward pass, and gradients remain available through the current-model activations. The anchor introduces no additional parameter freezing: MiniGPT retains full-parameter training, Qwen retains its configured last-six-layer training mode, and Llama retains LoRA-only training.

E.2 ANCHOR CALIBRATION

The reference scale in Eq. E.2 is computed from a fixed calibration set C containing 32 GEN and 32 MEM training examples, with the frozen reference model in evaluation mode.

Anchor strength is calibrated separately for each mechanistic case using 100-update pilot continuations with $\lambda_{\text{anchor}} \in \{0, 0.1, 1, 10\}$, where zero denotes the unanchored baseline. Each pilot starts from the same GEN-only endpoint with samples examples from the mixed training pool.

For a partition of C into calibration batches $C_1, \ldots, C_J$, the evaluation drift used for calibration is

$$\widehat{D}_F(C) = \frac{\sum_{j=1}^{J} |C_j| D_F(C_j)}{\sum_{j=1}^{J} |C_j|}, \tag{E.4}$$

where $|C_j|$ denotes the number of examples in calibration batch C_j, and all activations in this evaluation are computed in evaluation mode.

We select the smallest positive candidate coefficient whose post-pilot calibration drift is at most 25% of the unanchored pilot's drift, provided the baseline drift exceeds 10^{-12}. If no candidate satisfies this criterion, we use $\lambda_{\text{anchor}} = 10$ and record the calibration as ineffective. Full anchored training then restarts from the original GEN-only endpoint.

During training, the frozen reference and current model both run in training mode on the same sampled inputs using paired dropout RNG states. Reference parameters remain frozen, but reference activations are recomputed for each sampled microbatch. Calibration scales and evaluation drift are computed in evaluation mode.

Table E.1: Selected activation-anchoring strengths.

Case	Selected λ_{anchor}
MiniGPT–Addition	1
Qwen-0.8B–Addition	0.1
Llama-LoRA–Logical Deduction	10

E.3 Activation-drift evaluation

Activation drift is evaluated using a fixed calibration set of 32 GEN- and 32 MEM-training examples. Reference scales are computed once and remain fixed throughout continuation. Following the implementation used for these experiments, batch-level drift values are aggregated by the number of examples in each evaluation batch:

$$D_{\text{reported}} = \frac{\sum_B |B| D(B)}{\sum_B |B|}. \tag{E.5}$$

To compare cases with different raw activation scales, we additionally report the fraction of drift remaining relative to ordinary GEN-first mixed continuation:

$$R_{\text{drift}} = \frac{D_{\text{anchor}}}{D_{\text{Mix}}} \times 100\%. \tag{E.6}$$

Table E.2: GEN-family activation drift under ordinary mixed continuation and anchoring.

Case	Ordinary Mix drift	GEN-family anchor drift	Remaining drift
MiniGPT	8.8535	0.01385	0.16%
Qwen	1.6585	0.11165	6.73%
Llama-LoRA	0.12426	0.001491	1.20%

The reported percentages refer specifically to the normalized activation-drift metric in Equation E.3. They should not be interpreted as the fraction of a representation that changed or as a direct measure of functional preservation.

Table E.3 summarizes the final drift reduction, accuracy, and post-training ablation results for the three activation-anchoring cases.

Table E.3: Activation anchoring and post-training ablation. Anchoring strongly suppresses GEN-family activation drift while allowing high MEM acquisition. Ablating the same family afterward substantially reduces MEM. Accuracies and drops are reported in percentage points.

Setting	Drift reduction	GEN	MEM	MEM drop after ablation
MiniGPT–Addition	99.84%	53.0	97.5	33.0
Qwen-0.8B–Addition	93.27%	60.5	100.0	100.0
Llama-3.2-3B-LoRA–Logical Deduction	98.80%	93.0	98.0	91.5

E.4 POST-ANCHOR FUNCTIONAL DEPENDENCE

After anchored continuation, we ablate the same fixed GEN family and reevaluate both objectives on the confirmation sets.

Table E.4: Post-anchor same-family ablation results. Values are percentages.

Case	Anchored GEN	Ablated GEN	GEN drop	Anchored MEM	Ablated MEM	MEM drop
MiniGPT	53.0	42.0	11.0 pp	97.5	64.5	33.0 pp
Qwen	60.5	0.0	60.5 pp	100.0	0.0	100.0 pp
Llama-LoRA	93.0	14.0	79.0 pp	98.0	6.5	91.5 pp

These results distinguish measured activation drift in the fixed neuron family during learning from the later functional dependence of the learned behavior on that family. They do not establish that MEM is represented exclusively within the GEN family or that the selected family constitutes a complete MEM circuit.

E.5 ACTIVATION-ANCHORING ENDPOINTS

Section 4.4 constrains activation drift in the fixed GEN-associated neuron family during GEN-first mixed continuation. Table E.5 reports the corresponding endpoint quantities. Drift is measured relative to the frozen GEN-only reference using the normalized activation-drift measure in Equation E.3.

Table E.5: Activation-anchoring endpoint results. Drift reduction is relative to ordinary GEN-first mixed continuation. GEN and MEM are final anchored accuracies (%); MEM drop is the decrease in MEM accuracy after ablating the same fixed family at the anchored endpoint.

Setting	Drift reduction	GEN	MEM	MEM drop (pp)
MiniGPT–Addition	99.84%	53.0	97.5	33.0
Qwen-0.8B–Addition	93.27%	60.5	100.0	100.0
Llama-3.2-3B-LoRA–Logical Deduction	98.80%	93.0	98.0	91.5

Across all three cases, anchoring suppresses measured activation drift by more than 93% while preserving high MEM acquisition. Ablating the same family afterward nevertheless produces substantial MEM losses, providing the endpoint quantities underlying the temporal analysis in Section 4.4.