# STABILITY DOES NOT MIRROR PLASTICITY: A MODULE-LEVEL VIEW OF DECLARATIVE AND PROCEDURAL LEARNING IN TRANSFORMERS

**Jinjun Peng***, **Weiliang Zhao***, **Baishakhi Ray, Zhou Yu, Junfeng Yang**
Department of Computer Science
Columbia University
{jinjun,weiliang}@cs.columbia.edu

## ABSTRACT

Plasticity and stability in continual learning are typically studied at the entire-model level and in a task-agnostic manner, yet prior work has reported divergent conclusions across settings. Inspired by cognitive distinctions between *declarative* and *procedural* memory, we instead study how plasticity and stability vary across transformer modules and task types. In a controlled synthetic environment with clean measurement, we show a double dissociation: query-key (QK) and value-output (VO) projections are the most plastic on procedural tasks, whereas MLPs are the most plastic on declarative memorization. Stability, however, does not simply mirror plasticity: when learning a new procedure, updating QK retains over half of the stored declarative memory, while updating MLPs retains far less and updating VO loses roughly four-fifths. Targeted module reset-and-recovery experiments explain this asymmetry: plasticity tracks which module is best suited to a task's primary computation or storage, whereas stability depends on all ways the existing capability relies on that module, including the access to information stored in other modules. Experiments on a realistic model and tasks broadly echo these asymmetric dynamics. Overall, our work shows that plasticity and stability depend on different aspects of a module's functional role, offering a task-dependent, module-level perspective on continual learning.

## 1 INTRODUCTION

Existing continual learning study predominantly frames the plasticity (the ability to acquire new capabilities) and stability (the preservation of previously acquired knowledge) trade-off at the *entire-model* level and in a *task-agnostic* manner, seeking universal recipes including LoRA (Hu et al., 2022) and self-distillation (Yang et al., 2024; Shenfeld et al., 2026). However, their empirical findings vary across adaptation settings. Across tasks, while Biderman et al. (2024) evaluate code and math observing LoRA "learns less and forgets less," Pletenev et al. (2025) find that injecting factual knowledge through LoRA still lead to noticeable degradation on external question-answering benchmarks. Across architectural modules, Schulman & Lab (2025) show that attention-only LoRA underperforms MLP-only LoRA, whereas Zhu et al. (2025) demonstrate that tuning only self-attention projections allows multimodal models to acquire new downstream skills with almost no forgetting. We argue that these varied findings motivate moving beyond a model-level, task-agnostic perspective: plasticity and stability should be characterized at the level of individual task families and transformer modules, in relation to the functional roles each module plays in each task.

To systematically disentangle both tasks and architectural modules, cognitive science provides a foundational blueprint. Decades of memory research establish a classic distinction between two fundamental systems: *declarative* memory ("knowing that"), which supports factual, associative knowledge recall; and *procedural* memory ("knowing how"), which governs multi-step rule execution and algorithmic skills (Cohen & Squire, 1980; Squire, 1992). Neuropsychological studies reveal a double dissociation between them, suggesting that declarative and procedural memory de-

* Equal Contribution

pend on at least partially distinct neural substrates: amnesic patients acquire procedural skills such as mirror reading despite having no recollection of the training episodes (Cohen & Squire, 1980), whereas patients with neostriatal dysfunction show the reverse pattern, failing to learn a habit-like classification task while still remembering the training episodes (Knowlton et al., 1996). Modern transformers exhibit an analogous distinction along both functional and structural dimensions. Functionally, language models are simultaneously required to serve as vast factual encyclopedias (declarative tasks) and multi-step operational executors (procedural tasks). Structurally, mechanistic interpretability research suggest an analogous functional specialization: feed-forward layers act as key-value stores for factual associations (Geva et al., 2021; Meng et al., 2022), whereas attention circuits route information across positions and implements algorithmic computations (Elhage et al., 2021; Olsson et al., 2022). However, existing research examines this heterogeneity primarily under static conditions, without addressing how it affects the dynamic trade-off between plasticity and stability during continual adaptation. Crucially, as our experiments show later, this relationship is not a simple binary mapping in which each module exclusively serves and affects one task family.

To rigorously analyze these dynamics, standard pre-trained models are too noisy: task boundaries are often ambiguous, prior data exposure is unknown, and the amount of memorized information is difficult to quantify independently of generalization. To remove these confounds, we therefore design a clean synthetic testbed with two archetypal task families. For declarative memory, we use random sequence memorization and quantify the amount of memorization in bits, following the information-theoretic framework of Morris et al. (2026). For procedural task, we utilize algorithmic graph traversal (BFS and DFS) evaluated by sequence-level accuracy. Under a two-stage learning protocol, we evaluate both new-task plasticity and old-task stability across transformer modules.

Our synthetic experiments reveal a double dissociation: QK and VO projections are the most plastic on procedural tasks, whereas MLPs are the most plastic on declarative memorization. More importantly, cross-task stability does not simply follow or reverse plasticity. When learning a new procedure, updating QK retains over half of the stored declarative memory, whereas updating MLPs retains far less, and updating VO even loses roughly four-fifths, though VO and QK are equally plastic on the procedure itself. We further probe the source of this asymmetry through targeted module reset-and-recovery experiments. They show that procedural skill is primarily implemented in QK and VO, and declarative information is preferentially stored in MLPs, matching where each module is most plastic; yet QK and VO also support declarative memory by providing access to storage held in MLPs. Updating a module can therefore impair a capability it does not primarily implement: VO, though a weaker storage site than MLPs, can disrupt declarative retention more severely.

Finally, we study a pretrained model (Qwen2.5-0.5B) on various realistic procedural and declarative tasks, showing that our findings largely carry over. For instance, on the code execution task (procedural), QK-tuning achieves 29.4% accuracy while retaining over 67.2% of factual QA capability (declarative), whereas VO, MLP, and full tuning achieve similar or lower procedural performance but retain only 33.0%, 6.8%, 4.6% declarative knowledge. Together, these results indicate that plasticity and stability depend on different aspects of a module's functional role in a task, and therefore exhibit asymmetric patterns. Our primary contributions are summarized as follows:

- We frame continual learning at the level of task families and transformer modules, through the lens of cognitive memory systems (declarative vs. procedural) and their specialized neural substrates.
- We design a synthetic testbed that cleanly measures declarative memory in bits and procedural skill by exact execution, revealing a double dissociation in plasticity and an asymmetric, non-mirrored pattern in cross-task stability.
- Through targeted reset-and-recovery experiments, we provide empirical evidence that these properties depend on different aspects of a module's functional role: plasticity tracks a module's suitability for implementing a task's primary mechanics, whereas stability additionally reflects auxiliary roles such as access to storage held elsewhere, rather than a strict task-modular division.
- We show these findings largely carry over to fine-tuning a pretrained model on real-world procedural and declarative tasks, where modules reaching similar procedural performance can differ sharply in how much declarative knowledge they retain.

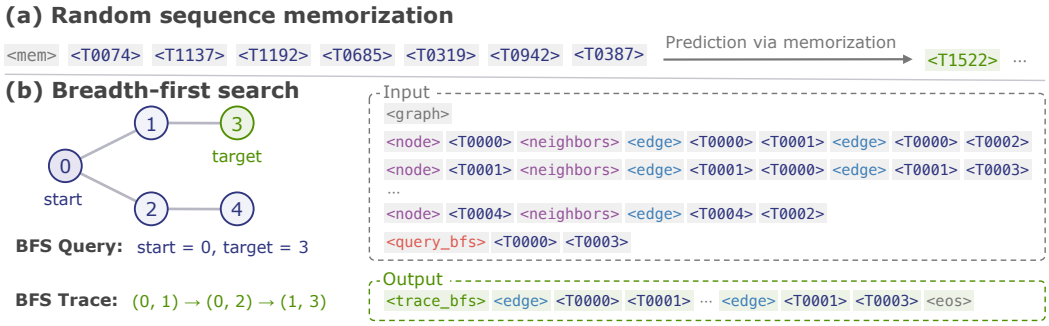



Figure 1: (a) Synthetic Declarative Task and (b) Procedural Task

## 2 Synthetic Testbed and Results

To isolate the plasticity and stability dynamics from the confounds of pre-training data distributions, ambiguous task boundaries, and unquantifiable parametric memory, we build a testbed following the controlled synthetic approach of Allen-Zhu (2024). The testbed separates the memorization of arbitrary information from the execution of procedural rules, allowing us to measure declarative memory and procedural skill independently and quantitatively. In this section, we first introduce the two task families and their measurements, then describe the continual learning protocol used to conduct experiments, and finally analyze the observed learning and retention dynamics.

### 2.1 Declarative and Procedural Tasks with Measurements

**Declarative Task: Random Sequence Memorization** To isolate information storage from the learning of generalizable rules, we define a fixed memory bank $\mathcal{D}_{\text{mem}} = \{x^{(i)}\}_{i=1}^{N}$, where each sequence $x^{(i)}$ consists of $L$ tokens sampled independently and uniformly from a discrete vocabulary $\mathcal{V}$ of size $V$. We set $L = 64$ and $V = 2048$. The model learns these sequences through standard autoregressive next-token prediction. Because tokens are sampled independently, the generating distribution provides no structural or syntactic patterns that predict their continuation. Successful recall therefore depends on learning sequence-specific associations from the memory bank, making this task a controlled proxy for declarative memory. Following Morris et al. (2026), we quantify memorization as the reduction in description length relative to a uniform reference:
$\text{mem}_U(\theta; \mathcal{D}_{\text{mem}}) = \sum_{x \in \mathcal{D}_{\text{mem}}} \max\{0, (L-1)\log_2 V + \sum_{t=2}^{L} \log_2 p_\theta(x_t \mid x_{<t})\}$,
where predictions are conditioned on ground-truth prefixes, with the shared task prefix omitted from the notation. $\text{mem}_U$ quantifies how much memorized declarative information about $\mathcal{D}_{\text{mem}}$ can be read out from the current model $\theta$, measured in bits. Figure 1(a) shows this task.

**Procedural Task: Algorithmic Graph Traversal** To test the execution of procedural tasks, we use breadth-first search (BFS) and depth-first search (DFS). For each task $\tau \in \{\text{BFS}, \text{DFS}\}$, we construct training and test sets $\mathcal{D}_\tau^s = \{((G_i, q_i), y_i)\}_{i=1}^{N_\tau^s}$, where $s \in \{\text{train}, \text{test}\}$. As shown in Figure 1(b), each input provides a connected undirected graph $G_i$ and a traversal query $q_i$ in-context, and the model must generate the corresponding traversal trace $y_i = (e_{i,1}, \ldots, e_{i,T_i})$ as a sequence of edges. Each graph contains 32 nodes with maximum degree 3, and thus the trace contains $T_i \in [2, 31]$ edges. For each task, we use $N_\tau^{\text{train}} = 256\text{k}$ training samples and $N_\tau^{\text{test}} = 10\text{k}$ test samples. The two tasks share the same graph representation and output format but differ in how they organize traversal, featuring two different execution procedures to learn. They require the model to track the evolving traversal state differently and select each subsequent step accordingly. To assess procedural skill acquisition beyond memorizing training examples, we evaluate on held-out graphs. Our primary metric is sequence-level exact-match accuracy under autoregressive greedy decoding: $\text{Acc}_\tau(\theta; \mathcal{D}_\tau^{\text{test}}) = \sum_{((G,q),y) \in \mathcal{D}_\tau^{\text{test}}} \mathbb{I}[\hat{y}_\theta(G, q) = y] / N_\tau^{\text{test}}$.

The two tasks thus distinguish declarative memory of arbitrary observed content from procedural skill of executing on unseen inputs. Their respective measurements provide the basis for evaluating new-task learning and old-task retention under the continual learning protocol below.

### 2.2 CONTINUAL LEARNING PROTOCOL

Unlike training from random initialization on a single task, continual learning requires adapting a model that already possesses knowledge and skills. We therefore use a two-stage protocol to measure plasticity and stability from an established starting point, where new learning and its effects on prior capabilities can be observed together.

**Stage 1: Establishing Declarative Memory and Procedural Skill** We use the "8L512D" transformer (25M parameters) from Allen-Zhu (2025) in all synthetic experiments. Starting from random initialization, we first train the full model on $\mathcal{D}^{\text{old}}_{\text{mem}} \cup \mathcal{D}^{\text{train}}_{\text{BFS}}$ to establish both declarative memory and procedural skill. The resulting checkpoint, denoted by $\theta_1$, achieves $\text{Acc}_{\text{BFS}}(\theta_1; \mathcal{D}^{\text{test}}_{\text{BFS}}) = 93.25\%$ and a memory density of $\text{mem}_U(\theta_1; \mathcal{D}^{\text{old}}_{\text{mem}})/P = 3.60$ bpp (bits per parameter), where $P$ is the total parameter count. This memory density matches the 3.6 bpp reported by Morris et al. (2026) because we set $|\mathcal{D}^{\text{old}}_{\text{mem}}| = 196\text{k}$ based on their estimation to fully saturate the model, ensuring substantial memorization alongside procedural competence. We use this checkpoint as the common starting point for all Stage-2 adaptation experiments. By default, all results are averaged over experiments with three random seeds.

**Stage 2: New-Task Adaptation and Cross-Task Retention** Starting from $\theta_1$, we train to obtain $\theta_2$ in two separate adaptation setups:

- **Procedural Adaptation**: we train on $\mathcal{D}^{\text{train}}_{\text{DFS}}$, measuring plasticity by $\text{Acc}_{\text{DFS}}(\theta_2; \mathcal{D}^{\text{test}}_{\text{DFS}})$ and stability by the retention ratio $\text{mem}_U(\theta_2; \mathcal{D}^{\text{old}}_{\text{mem}})/\text{mem}_U(\theta_1; \mathcal{D}^{\text{old}}_{\text{mem}})$.
- **Declarative Adaptation**: we train on $\mathcal{D}^{\text{new}}_{\text{mem}}$ disjoint from $\mathcal{D}^{\text{old}}_{\text{mem}}$, measuring plasticity by $\text{mem}_U(\theta_2; \mathcal{D}^{\text{new}}_{\text{mem}})$ divided by trainable parameter count in bpp and stability by the retention ratio $\text{Acc}_{\text{BFS}}(\theta_2; \mathcal{D}^{\text{test}}_{\text{BFS}})/\text{Acc}_{\text{BFS}}(\theta_1; \mathcal{D}^{\text{test}}_{\text{BFS}})$.

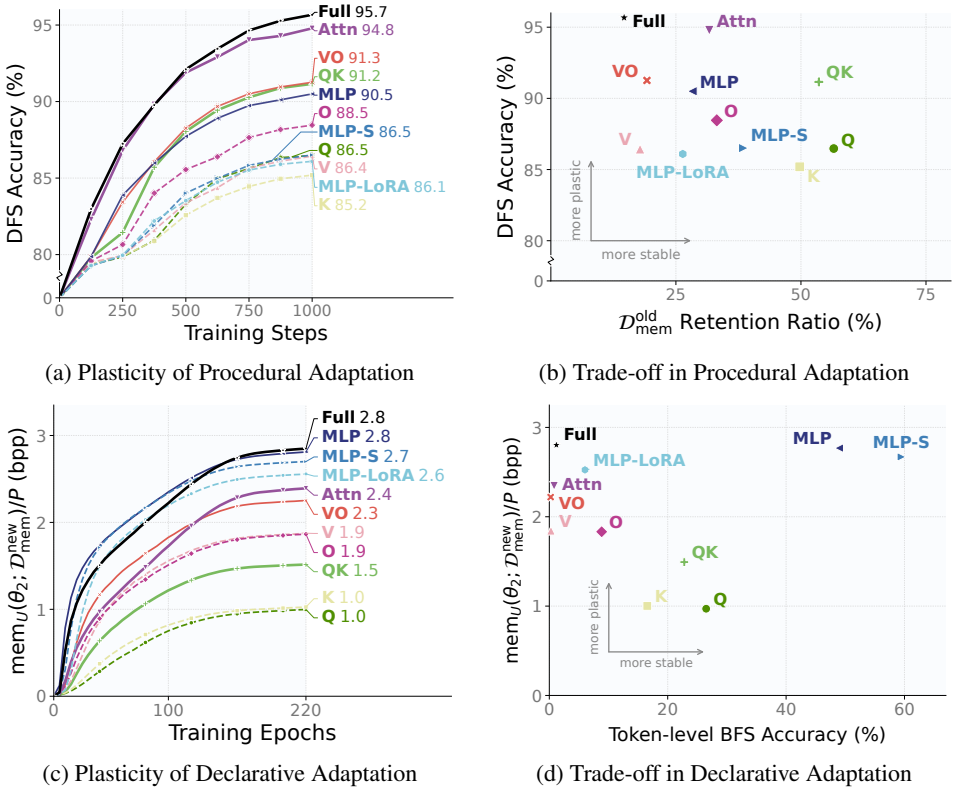
We deliberately focus on stability across task families: declarative memory after learning a new procedure, and procedural skill after learning new declarative information, to investigate whether learning for one type of capability interferes with a qualitatively different capability already present in the model. This axis is not directly captured by standard aggregate or within-task forgetting measurements in prior work, yet it is particularly informative for revealing that plasticity and stability follow different aspects of a module's functional role, as we show later.

**Module-Wise Adaptation** To study module-level dynamics, during Stage 2 we freeze all parameters except a targeted transformer module to update. We consider the query and key projections jointly as **QK**, the value and output projections jointly as **VO**, the complete self-attention block as **Attention**, and the feed-forward layers as **MLP**. We additionally report **Full** adaptation as a reference in which all model parameters are trainable. Because QK, VO have 4.2M parameters respectively, and MLP has 15.7M parameters, raw comparisons can otherwise confound architectural module type with trainable parameter count. We therefore include two trainable-parameter-matched MLP setups matching QK or VO parameter count: (1) **MLP-LoRA**: We apply rank-97 LoRA to all projections in every MLP layer, yielding 4.17M trainable parameters while freezing the original weights. (2) **MLP-S**: Since LoRA can produce distinct adaptation behavior, such as the emergence of intruder dimensions (Shuttleworth et al., 2025), we also directly update a randomly sampled subset of MLP channels, yielding 4.19M trainable parameters.

### 2.3 RESULTS IN SYNTHETIC TESTBED

**Plasticity Shows a Task–Module Double Dissociation** We first compare how effectively different modules acquire a new capability from the common Stage-1 checkpoint. As shown in Figure 2a, when learning the new procedural task DFS, QK and VO learn faster, and achieve higher DFS accuracy (about 91.2%) than MLP (90.5%) and parameter-count-matched MLP setups (86.1-86.5%). Conversely, on declarative adaptation (learning $\mathcal{D}^{\text{new}}_{\text{mem}}$), this plasticity order reverses. In Figure 2c comparing the capability of acquiring new declarative information, MLPs exhibit the highest plasticity on par with full-parameter learning , reaching 2.80 bpp at full capacity, 2.70 bpp with MLP-S,

Even full model training cannot reach about 3.6 bpp anymore, possibly due to the loss of plasticity (Dohare et al., 2024)




(a) Plasticity of Procedural Adaptation

(b) Trade-off in Procedural Adaptation

(c) Plasticity of Declarative Adaptation

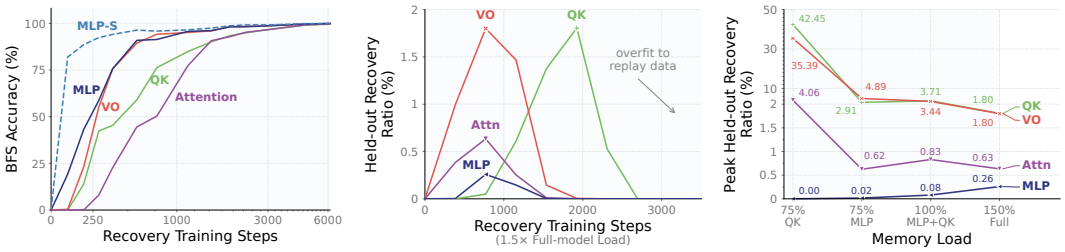
(d) Trade-off in Declarative Adaptation

Figure 2: Synthetic experiments show asymmetric plasticity and cross-task stability.

and 2.60 bpp with MLP-LoRA, compared with 2.30 bpp for VO and 1.54 bpp for QK. These results reveal a double dissociation in plasticity: QK and VO are more effective for learning a new procedure, whereas MLP is more effective for acquiring new declarative information. Thus each task family favors different modules for acquiring a new capability.

**Stability Exhibits a Distinct Task–Module Pattern** We next compare how well previously acquired capabilities survive new-task learning. Figure 2b shows that when learning the new procedure DFS, updating different modules exhibits markedly divergent stability of previously acquired declarative memory. Updating QK is more stable, preserving about 54% of the declarative information memorized in stage-1, compared with 28% for MLP, 26% for MLP-LoRA and 39% for MLP-S. By contrast, when learning new declarative information, the stability order of previously learned procedural skill is different. Under greedy autoregressive decoding, accumulating single-step prediction errors causes sequence-level accuracy to drop to zero across all module setups after declarative learning converges. However, inspecting teacher-forced token accuracy reveals differences in Figure 2d: MLP-S and MLP retains 59.5%, 49.0% token accuracy, respectively, higher than 22.8% for QK and almost zero for VO. Thus, stability also varies sharply across modules and task directions, but its pattern is distinct from the plasticity's above.

**Cross-task Stability Does Not Mirror Plasticity** A natural hypothesis is that stability is the flip side of plasticity: a module that absorbs the new task family easily, or that has little affinity for the old one, should leave the old capability intact. However, our results above contradict this. First, similar plasticity does not imply similar stability: VO and QK have nearly identical plasticity in learning DFS, yet updating VO retains only about 19% of the stored declarative memory, far below the 54% retained by QK. Second, lower plasticity on a task family does not mean higher stability on the same family: VO stores new declarative information less effectively than MLPs (2.30 vs. 2.60–2.80 bpp), yet updating VO for DFS erases more of the old declarative memory than MLPs (19% vs. 26–39%). Together, these results show that plasticity and cross-task stability cannot be reduced to a single mirrored module ordering. This asymmetry motivates the targeted interventions




(a) BFS Recovery Process (b) Memory Recovery Process (c) All-load Memory Recovery

Figure 3: Module-wise Reset-and-Recovery on Procedural Skill and Declarative memory

in the next section, where we examine which aspects of a module's functional role are associated with each property.

## 3 Functional Attribution via Targeted Reset-and-Recovery

The asymmetries above suggest that plasticity and stability may follow different aspects of how a module contributes to a capability. However, attributing a capability to an individual module in a naturally full-parameter-trained transformer is nontrivial. Removing any single module destroys both capabilities, every module can implement either task family to some extent given sufficient training, and $\mathrm{mem}_U$ cannot be decomposed into per-module storage. We therefore use an intervention that asks how central a module is to a task family, rather than whether it is necessary. We reinitialize one module of a trained model each time, retrain the full model, and track how the task capability recovers. If the reset module plays an auxiliary role, the core implementation survives in the remaining modules, and thus the capability will be restored quickly. If the module plays a primary role, the capability must be rebuilt, or the information it held is lost. As we show below, this intervention separates a module's primary role in a capability from supporting roles.

### 3.1 Procedural Skill Is Primarily Implemented in QK and VO

Starting from $\theta_1$, we reset QK, VO, Attention, or MLP and retrain the full model on $\mathcal{D}^{\mathrm{train}}_{\mathrm{BFS}}$. All setups eventually return to near-perfect BFS accuracy, so we compare early recovery efficiency rather than the final plateau. As shown in Figure 3a, resetting MLP recovers fastest, resetting VO is next, resetting QK is more slowly, and resetting the entire attention block is slowest. Resetting a parameter-matched random subset of MLP channels leaves BFS largely intact and recovers the fastest, confirming that the gap is not explained by the number of reset parameters. Also, MLPs are not simply faster procedural learners themselves, as Section 2.3 shows MLPs learn procedural tasks less efficiently than Attention, QK and VO. Together, these observations suggest that the fast recovery after an MLP reset reflects procedural skills surviving in the remaining modules, whereas resetting QK or VO removes more of the learnt procedure, which must then be rebuilt, thus leading to slower convergence. This indicates that procedural skill is primarily implemented in the attention projections, whereas MLPs contribute in a supporting role that is easier to restore, matching where procedural plasticity is highest in Section 2.3.

### 3.2 Declarative Memory Relies on MLP Storage and Attention Access

To probe declarative memory, we start from a checkpoint $\theta_m$ sufficiently trained only on $\mathcal{D}_{\mathrm{mem}}$, reset one module, and retrain the full model on a 10% subset of $\mathcal{D}_{\mathrm{mem}}$ as replay to obtain $\theta^r_m$. We then measure $\mathrm{mem}_U$ on the remaining 90% that is never replayed. Unlike procedural skill, arbitrary sequences cannot be relearned without re-exposure, so held-out information can reappear only if it survived in the remaining modules, with replay merely restoring the auxiliary function (e.g., access or readout existing storage) of the reset module. If the reset module itself held some information, the corresponding $\mathrm{mem}_U$ should stay near zero. Because every module can store declarative information (as shown in Figure 2c) to some extent, we vary the memory load based on the 3.6 bpp estimation confirmed in Section 2.2: 75% of the maximum capacity of QK (so theoretically

QK or VO alone can fully store), 75% of that of MLP (theoretically MLP can fully store, whereas QK+VO cannot), 100% of the combined MLP+QK capacity (where MLP alone no longer suffices), and $1.5\times$ that of the full model (the same as $\mathcal{D}_{\text{mem}}^{\text{old}}$, where no standalone module can fully store). We report the peak recovered ratio $\text{mem}_U(\theta_m^r; \mathcal{D}_{\text{mem}}^{\text{held-out}}) / \text{mem}_U(\theta_m; \mathcal{D}_{\text{mem}}^{\text{held-out}})$, since prolonged training eventually overfits the replay subset and erases all held-out information, as shown in Figure 3b.

As shown in Figure 3c, the four loads reveal distinct contributions of storage and access to declarative memory. Under the two lighter loads, resetting MLP leaves essentially no recoverable held-out memory (0.00% and 0.02%). This holds even under the 75% QK load, where QK or VO alone has sufficient capacity to hold the entire memory bank, potentially allowing MLP reset-and-recovery to read it out. By contrast, resetting QK or VO leaves part of the held-out memory recoverable: 42.45% and 35.39% under the 75% QK load, and 2.91% and 4.89% under the 75% MLP load, respectively. Together, these contrasting results suggest that MLPs are the preferred storage site for declarative information when their capacity suffices, whereas QK and VO serve mainly to *access* the storage. Limited replay cannot restore the information lost with the reset module but can partially restore the access to information, allowing information preserved in the remaining modules to become readable again. This echoes prior findings that sparse replay can mitigate forgetting in language models (de Masson d'Autume et al., 2019; Scialom et al., 2022; Zheng et al., 2025).

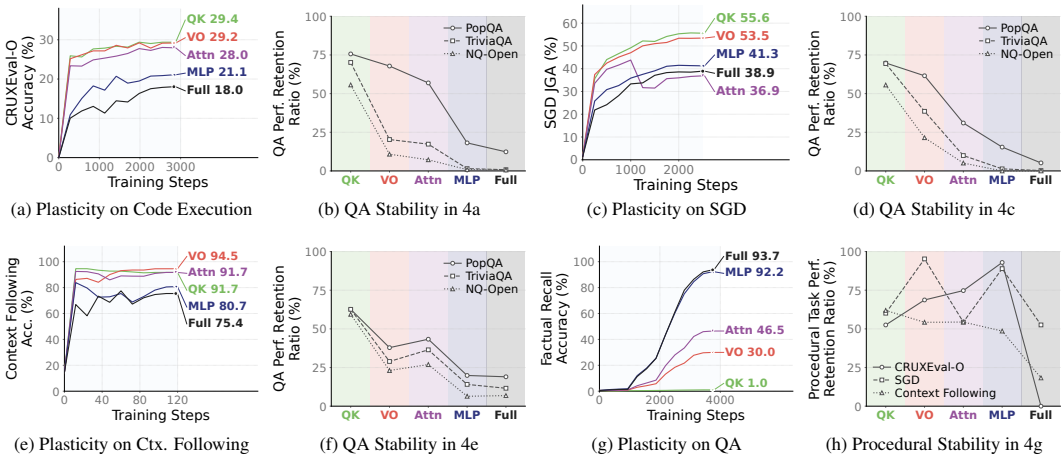
Recovery after resetting QK or VO declines as the load grows, falling from 35-42% to 3–5% and then to about 1.8% under the $1.5\times$ full-model load, likely because access to a larger declarative memory is more difficult to recover through replay. Against this trend, recovery after resetting MLP increases once the memory load exceeds MLP capacity: it is nearly zero under the two lighter loads, becomes small but positive (0.08%) at the 100% MLP+QK load, and reaches 0.26% under the highest load. These opposite trends suggest that the increase under MLP-reset-and-recovery is not simply due to heavier loads making reset-and-recovery easier. Instead, they indicate that once the memory load exceeds what MLPs can hold, QK and VO also take on part of the storage, and that part can survive an MLP reset. Thus, storage and access are not an exclusive assignment of roles to modules, but different aspects of a module's functional contributions, and declarative memory depends on both.

### 3.3 Connecting Functional Roles to Plasticity and Stability

These interventions suggest that plasticity and stability relate to different aspects of a module's functional role (Section 2.3). Plasticity aligns with which module is best suited to implementing a task's primary mechanics. Attention projections, which primarily implement procedural execution, are the most plastic in procedural learning. MLPs, the preferred storage site for declarative information, are the most plastic in declarative learning. Stability, however, reflects the broader set of roles a module plays in an existing capability, including supporting roles beyond where the capability is preferentially implemented. Updating a module can therefore interfere not only with information directly stored there, but also with the access to information stored elsewhere. In particular, VO's poor declarative retention under procedural adaptation is consistent with disrupting memory access, despite its weaker storage contribution than MLPs. Its lower declarative retention than QK may reflect its more critical role in memory access, or greater vulnerability of that role to our specific procedural learning updates. This interpretation is further supported by mixing only 1% of $\mathcal{D}_{\text{mem}}^{\text{old}}$ into Stage-2 DFS procedural learning: while maintaining similar DFS performance, VO now retains 75.52% of the declarative memory, close to QK (80.14%) and well above MLP (56.34%). This reverses the no-replay ordering between VO and MLP (19% vs. 28%), supporting that VO mainly serves the access role, which is easier to preserve or restore than MLP's storage. Together, these findings indicate that plasticity tracks a module's suitability for implementing a task's primary mechanics, whereas stability reflects the full set of functional dependencies an existing capability has on the module, including supporting roles. Because where a capability is preferentially implemented does not fully capture these dependencies, plasticity and stability exhibit asymmetric rather than mirrored patterns.

## 4 Realistic Setups and Results

We now ask whether the same picture holds when a pretrained large language model is updated on realistic tasks. We use Qwen2.5-0.5B (Qwen Team, 2024) as the base model $\theta_{\text{p}}$ and, as in Stage 2 of Section 2.2, update a single module each time (QK, VO, Attention, or MLP) while freezing the




(a) Plasticity on Code Execution (b) QA Stability in 4a (c) Plasticity on SGD (d) QA Stability in 4c

(e) Plasticity on Ctx. Following (f) QA Stability in 4e (g) Plasticity on QA (h) Procedural Stability in 4g

Figure 4: Learning Curves and Respective Cross-Task Retention Levels on Realistic Tasks

rest, with Full as a reference, to obtain $\theta_{\text{ft}}$. Since our goal is to test whether the findings in previous sections transfer rather than to maximize continual learning performance, we deliberately choose a relatively weak model that leaves substantial headroom across tasks to observe plasticity, instead of recent stronger models that already saturate many of them.

**Realistic Tasks** We select three procedural tasks that require input-dependent operations whose outcomes cannot be recalled from in-weight knowledge: (1) **Code Execution**: given an in-context program and input, predict the output by simulating execution; we train on PYX (Ding et al., 2024) and evaluate output accuracy on CRUXEVAL-O (Gu et al., 2024). (2) **Dialogue-to-API Conversion**: given a user-assistant dialogue and an API schema, generate the API call reflecting the latest dialogue state; we train and evaluate on Schema Guided Dialogue (SGD) (Lee et al., 2022), and measure fuzzy joint goal accuracy (JGA). (3) **Contextual Information Following**: given a passage and a question, answer strictly according to the passage even when it conflicts with in-weight knowledge; we construct splits from the three QA sources below and measure strict-following accuracy. For declarative tasks, we use factual question answering focused on recall rather than complex reasoning, sampling 5k QA pairs each from PopQA (Mallen et al., 2023), NQ-Open (Lee et al., 2019), and TriviaQA (Joshi et al., 2017). Figure 6 shows these tasks.

**Adaptation and Evaluation Protocol** For procedural adaptation, we fine-tune $\theta_{\text{p}}$ separately on each procedural task, evaluate plasticity by the task's accuracy metric, and measure stability as the retention ratio on factual QA, i.e., the proportion of questions correctly answered by $\theta_{\text{p}}$ that remain correct after adaptation. For declarative adaptation, because $\theta_{\text{p}}$ has near-zero procedural accuracy, we first establish all three procedural skills through mixed training and then fine-tune on the 15k pooled QA pairs. Plasticity is strict closed-book factual recall on the trained QA pairs, measuring memorization rather than held-out generalization, and stability is the ratio of each procedural task's performance after to before QA learning.

**Plasticity Preferences Carry Over** The top row of Figure 4 reproduces the double dissociation. On all three procedural tasks, attention projections learn faster and reach higher accuracy than MLP: QK and VO reach 29.4% and 29.2% on CRUXEVAL-O versus 21.1% for MLP (Figure 4a), 55.6% and 53.5% JGA on SGD versus 41.3% (Figure 4c), and 91.7% and 94.5% context-following accuracy versus 80.7% (Figure 4e). The preference reverses for factual acquisition (Figure 4g): MLP reaches 92.2% recall, on par with Full (93.7%), whereas Attention, VO, and QK reach only 46.5%, 30.0%, and 1.0%. The dissociation is thus even sharper than in the synthetic testbed, with QK nearly unable to effectively memorize new facts.

Reasoning tasks such as math can be viewed as a composite of both families: the model must recall axioms, theorems, and learned solution strategies (declarative), then apply input-specific operations to the given problem (procedural).

**Cross-Task Stability Remains Asymmetric** As in Section 2.3, similar plasticity does not imply similar stability. Although QK and VO reach close procedural accuracy (Figures 4a, 4c and 4e), the retention panels (Figures 4b, 4d and 4f) demonstrate that QK retains far more factual QA than VO (e.g., 70.2% vs. 20.3% on TriviaQA after learning code execution), and the QA stability ordering QK $>$ VO $>$ MLP $>$ Full holds across all procedural tasks learning. This matches the access-disruption signature in Section 3: VO is not the primary declarative storage site, yet updating it disrupts declarative knowledge far more than updating QK. Unlike in the synthetic testbed, updating MLP here loses more declarative knowledge than updating VO. A plausible reason is that realistic procedural tasks are not purely procedural, still drawing on in-weight knowledge, so procedural training may contain implicit declarative replay that preserves VO's access role, whereas facts overwritten in MLP storage cannot be restored this way. This matches the synthetic setting with 1% declarative replay, where VO retains more than MLP (Section 3.3). In the reverse direction (Figure 4h), updating MLP lets the model acquire far more trained facts while retaining most procedural task performance, echoing the synthetic finding that MLP updates best preserve procedural skill that is implemented mainly in attention projections (Section 3). Context following is the exception, plausibly because it requires answering from the passage even when it contradicts the model's own knowledge, and newly memorized facts can strengthen exactly this competing knowledge.

## 5 Limitations and Future Work

Our work provides a first view of plasticity and stability through module function, while both the architectural and functional dimensions can be studied at a finer resolution.

**Finer architectural granularity.** We treat QK, VO, and MLP as units of analysis, although each spans many layers and heads that may play different roles. Prior work already reveals submodule specialization: induction behavior is associated with particular attention heads and depths (Olsson et al., 2022), MLP functions vary across layers (Geva et al., 2021), and factual recall is mediated strongly by middle-layer MLPs (Meng et al., 2022). These findings suggest that the module-level roles we observe may themselves be distributed unevenly across finer components. Extending our analysis to individual layers, heads, and circuits could reveal whether storage, access, and procedural computation are concentrated in distinct subcomponents, and whether these subcomponents exhibit different plasticity–stability dynamics. This could support more selective adaptation that updates the components most useful for a new task while preserving those important to existing capabilities.

**Finer functional granularity.** We use declarative and procedural memory as archetypal functional demands rather than a complete quantitative taxonomy. Realistic tasks may rely on both to different degrees, motivating a characterization of task composition along at least two continuous dimensions: the amount of stored information that must be retrieved and the amount of input-conditioned computation that must be performed. In our synthetic setting, declarative load can be varied and quantified in bits relative to model capacity (Morris et al., 2026), whereas procedural tasks lack a comparable notion of load or capacity. Work that expresses explicit algorithms as transformer computations suggests possible resource measures such as depth and head count (Weiss et al., 2021; Lindner et al., 2023), although these do not yet quantify how much capacity a learned procedure occupies in a model. Developing such measures would allow procedural load to be varied systematically as the declarative load, and test how task complexity and composition shape plasticity and stability.

Overall, pursuing these finer structural and functional resolutions may ultimately enable adaptation strategies that allocate updates according to *where* and *what* a model needs to learn, supporting more efficient, continually evolving AI systems.

## 6 Conclusion

We show that different transformer modules specialize differently in acquiring and preserving declarative and procedural capabilities, and that plasticity does not mirror stability. Attention projections are most plastic for procedural learning, whereas MLPs are most plastic for declarative memorization, yet modules with similar plasticity can differ sharply in retention. Reset-and-recovery experiments indicate that plasticity follows a module's suitability for a task's primary computation or storage, while stability depends on the broader roles that existing capabilities rely on, including

access to information stored elsewhere. These asymmetric dynamics largely carry over to realistic tasks, motivating continual learning methods that consider both task type and module function.

## 7 AI USE STATEMENT

We used AI to assist with polishing the paper writing, identifying suitable and accessible datasets for our experiments, and implementing code based on our research designs and ideas to conduct experiments. We have reviewed all AI-assisted work. We take full responsibility for all content and results of this work, including those produced with AI assistance.

## REFERENCES


Zeyuan Allen-Zhu. ICML 2024 Tutorial: Physics of Language Models, July 2024. Project page: https://physics.allen-zhu.com/.

Zeyuan Allen-Zhu. Physics of Language Models: Part 4.1, Architecture Design and the Magic of Canon Layers. In *Proceedings of the 39th Conference on Neural Information Processing Systems*, NeurIPS '25, 2025. Full version available at https://ssrn.com/abstract=5240330.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=aloEru2qCG.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019. URL https://arxiv.org/abs/1902.10486.

Neal J. Cohen and Larry R. Squire. Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210(4466):207–210, 1980. doi: 10.1126/science.7414331.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f8d2e80c1458ea2501f98a2cafadb397-Paper.pdf.

Yangruibo Ding, Jinjun Peng, Marcus J. Min, Gail Kaiser, Junfeng Yang, and Baishakhi Ray. Semcoder: training code language models with comprehensive semantics reasoning. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024. doi: 10.1038/s41586-024-07711-7. URL https://doi.org/10.1038/s41586-024-07711-7.

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5484–5495, 2021.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751/.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution, 2024. URL https://arxiv.org/abs/2401.03065.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models, 2023. URL https://arxiv.org/abs/2301.04213.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning, 2022. URL https://arxiv.org/abs/2110.04366.

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. URL https://arxiv.org/abs/2106.09685.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

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*, 114(13): 3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Barbara J. Knowlton, Jennifer A. Mangels, and Larry R. Squire. A neostriatal habit learning system in humans. *Science*, 273(5280):1399–1402, 1996. doi: 10.1126/science.273.5280.1399.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference, 2024. URL https://arxiv.org/abs/2309.10105.

Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. Sgd-x: A benchmark for robust generalization in schema-guided dialogue systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10938–10946, 2022. ISSN 2159-5399. doi: 10.1609/aaai.v36i10.21341. URL http://dx.doi.org/10.1609/aaai.v36i10.21341.

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

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: compiled transformers as a laboratory for interpretability. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric

memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995. doi: 10.1037/0033-295X.102.3.419.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

John Xavier Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G. Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much can language models memorize? In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bA6BgSbaUi.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Sergey Pletenev, Maria Marina, Daniil Moskovskiy, Vasily Konovalov, Pavel Braslavski, Alexander Panchenko, and Mikhail Salnikov. How much knowledge can you pack into a LoRA adapter without harming LLM? In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4309–4322, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.243. URL https://aclanthology.org/2025.findings-naacl.243/.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019. URL https://arxiv.org/abs/1811.11682.

John Schulman and Thinking Machines Lab. LoRA without regret. *Thinking Machines Lab: Connectionism*, September 2025. doi: 10.64434/tml.20250929. URL https://thinkingmachines.ai/blog/lora/.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.410. URL https://aclanthology.org/2022.emnlp-main.410/.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2601.19897.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 174627–174662. Curran Associates, Inc., 2025. doi: 10.52202/085713-5808. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ff541950d1e885af90f523571564a401-Paper-Conference.pdf.

Larry R. Squire. Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. *Journal of Cognitive Neuroscience*, 4(3):232–243, 1992. doi: 10.1162/jocn.1992.4.3.232.

Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers, 2022. URL https://arxiv.org/abs/2203.09795.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/weiss21a.html.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1028–1043, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.58. URL https://aclanthology.org/2024.acl-long.58/.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 67081–67146, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a774503daed55eb53c634847ae071ec7-Paper-Conference.pdf.

Zhen Zhu, Yiming Gong, Yao Xiao, Yaoyao Liu, and Derek Hoiem. How to teach large multimodal models new skills. *arXiv preprint arXiv:2510.08564*, 2025. doi: 10.48550/arXiv.2510.08564. URL https://arxiv.org/abs/2510.08564.

## A Related Work

**Plasticity and stability in continual learning.** The trade-off was first documented as catastrophic interference (McCloskey & Cohen, 1989) and is typically addressed at the model level with task-agnostic remedies such as regularization (Kirkpatrick et al., 2017) and replay (Rolnick et al., 2019; Chaudhry et al., 2019). For language models, common remedies include parameter-efficient adaptation (Hu et al., 2022) and self-distillation (Yang et al., 2024; Shenfeld et al., 2026), whose effects vary across tasks: LoRA forgets little on code and math (Biderman et al., 2024) yet degrades QA when injecting facts (Pletenev et al., 2025). Moreover, much measured forgetting may reflect lost access rather than lost knowledge: Kotha et al. (2024) attribute it to a shift in the task inferred by the model, while Zheng et al. (2025) show that lost task alignment can be restored with a small amount of old-task data. Loss of plasticity over long task sequences (Dohare et al., 2024) is orthogonal to the single-step setting we study. These works largely treat the model as a whole; in contrast, we localize the access/storage distinction to modules: updating an access module (VO) yields forgetting that light replay largely reverses, whereas updating a storage module (MLP) yields forgetting that it only partly reverses (Section 3.3).

**Module-restricted adaptation.** Studies that restrict fine-tuning to a subset of modules have typically emphasized one side of the plasticity–stability trade-off. Attention-only fine-tuning is sufficient to adapt vision transformers (Touvron et al., 2022) and can teach vision-language models new skills with little forgetting (Zhu et al., 2025), whereas feed-forward layers can be more effective targets for parameter-efficient learning (He et al., 2022; Schulman & Lab, 2025); LoRA and full fine-tuning can also converge to different solutions (Shuttleworth et al., 2025). Taken separately, these results can appear conflicting because they concern different properties and task types; we instead measure both plasticity and stability across two task families and explain the resulting pattern through the distinct roles played by each module.

**Functional specialization of transformer modules.** Static interpretability work localizes factual associations to feed-forward key-value memories (Geva et al., 2021; Meng et al., 2022), links induction-based in-context learning to attention circuits (Elhage et al., 2021; Olsson et al., 2022), and shows that attention heads extract stored facts (Geva et al., 2023). However, causal localization does not necessarily identify the best target for intervention: where causal tracing locates a fact does not predict which MLP layer is best to edit (Hase et al., 2023). Controlled synthetic testbeds isolate architectural capabilities under clean measurement (Allen-Zhu, 2025), while information-theoretic work quantifies memorization in bits per parameter (Morris et al., 2026). We build on these tools to ask a dynamic question: how specialization shapes learning and retention under module-wise updates. We probe module roles through reset-and-recovery, which perturbs module parameters and measures capability recovery after retraining, and use the declarative/procedural distinction from memory research (Cohen & Squire, 1980; Squire, 1992; McClelland et al., 1995; Knowlton et al., 1996) as the organizing lens.

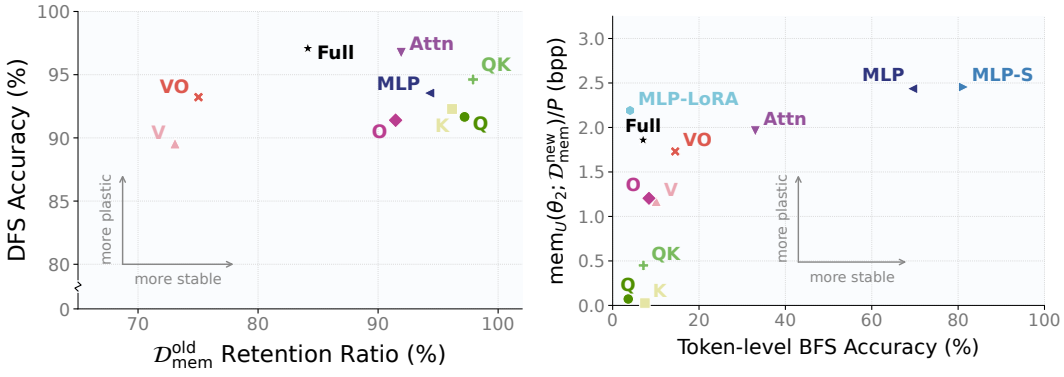
## B Reproducibility Statement

We ensure reproducibility by providing an anonymous repository (link) with source code, datasets, evaluation scripts used in this work.

## C Additional Details of Synthetic Experiments

**Training Setup** Unless otherwise specified, we use an effective batch size of 256 and optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with a weight decay of $0.01$. The learning rate is linearly warmed up over the first $2\%$ of training to a peak value of $3 \times 10^{-4}$, followed by cosine decay to $10\%$ of the peak learning rate. We apply gradient clipping with a maximum norm of $1.0$ and use `bfloat16` mixed-precision training.

**Results under a Non-Saturated Stage-1 Memory Load** Our main synthetic experiments deliberately use a heavily loaded Stage-1 checkpoint, where the declarative memory bank is chosen to




(a) Procedural adaptation

(b) Declarative adaptation

Figure 5: **Plasticity–stability trade-offs under a non-saturated Stage-1 memory load.** We reduce the Stage-1 declarative memory load to 50% of the estimated full-model capacity, at which the MLP alone is theoretically sufficient to store the entire memory bank. **(a)** DFS plasticity versus retention of the Stage-1 declarative memory after procedural adaptation. **(b)** New declarative memory acquired versus retained token-level BFS accuracy after declarative adaptation. The qualitative module-wise patterns closely match those observed under the saturated-load setting in the main text.

saturate the model's memorization capacity. To verify that the observed module-wise plasticity–stability patterns are not specific results of this saturation setup, we repeat both Stage-2 adaptation directions with a substantially lighter Stage-1 memory load, using only 50% of the estimated full-model capacity. At this load, the memory bank is small enough that the MLP alone is theoretically capable of storing all declarative information, removing the need to distribute storage across modules purely because of capacity constraints. As shown in Figure 5, the resulting trends closely match those in the main experiments. Under procedural adaptation, QK remains substantially more stable than VO while achieving comparable DFS performance, and MLP-based updates still preserve more memory than VO. In the reverse direction, when learning new declarative memory, MLP and MLP-S again provide the strongest combination of memory acquisition and BFS retention, whereas attention projections, particularly QK, acquire much less new memory despite preserving less of the previous procedural skill. Thus, the qualitative plasticity–stability asymmetries reported in the main text persist even when Stage 1 does not saturate model memory, suggesting that they reflect module-dependent functional roles regardless of the capacity saturation.

## D Additional Details of Realistic Experiments

Table 1 lists the number of samples used for training and evaluation in the realistic setting. The code-execution training rows are PyX programs whose target is the program output, without a reasoning trace. SGD training uses 20,000 user-turn service frames drawn from the official training split, and the SGD test set contains 11,539 frames of services seen during training and 34,577 frames of unseen services. The context-following training set contains 140 conflict passages whose answer has been substituted, 700 reading passages for questions the base model cannot answer closed-book, and 96 passages that agree with the model's knowledge, built from TriviaQA and SQuAD; its evaluation pairs each of the 1,683 PopQA facts the base model knows with a one-sentence passage that states a different answer of the same relation. The injected facts are 5,000 TriviaQA and 5,000 NQ-Open training questions and 4,966 PopQA questions; none overlaps an evaluation set, and the 33 facts the base model already answered correctly were removed. After injection, the SGD skill is measured on the 17,283 test frames the model solved before injection.

Below is the source of each dataset; the exact revisions and file hashes are recorded in the repository.

- PyX: `https://huggingface.co/datasets/semcoder/PyX`
- CRUXEval: `https://github.com/facebookresearch/cruxeval`

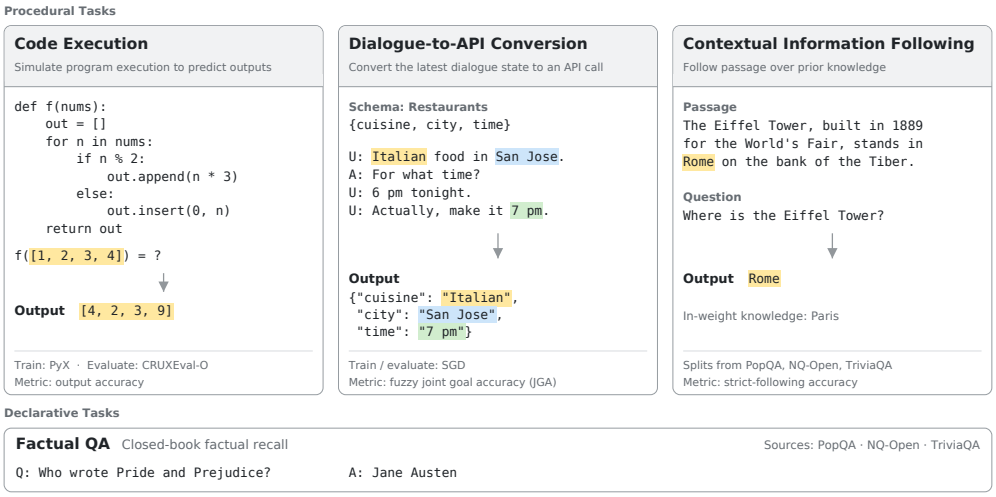



Figure 6: Realistic Procedural and Declarative Tasks

Table 1: Number of samples used for training and evaluation in the realistic setting. For the retention benchmarks, the count is the number of items the untouched base model answers correctly, out of the full benchmark.

| Task | Split | Data | Samples |
|---|---|---|---|
| Code execution | train | PyX | 22,666 |
| | eval | CRUXEval-O | 800 |
| Dialogue-to-API conversion | train | SGD user-turn service frames | 20,000 |
| | eval | SGD test frames | 46,116 |
| Context following | train | TriviaQA and SQuAD passages | 936 |
| | eval | PopQA conflict passages | 1,683 |
| Knowledge injection | train | TriviaQA, NQ-Open, and PopQA facts | 14,966 |
| | eval | injected facts, closed-book | 14,966 |
| | eval | SGD test frames | 46,116 |
| Knowledge retention | eval | PopQA | 1,683 / 14,267 |
| | eval | TriviaQA (validation) | 535 / 11,313 |
| | eval | NQ-Open (development) | 157 / 3,610 |

- SGD: `https://github.com/google-research-datasets/dstc8-schema-guided-dialogue`
- SQuAD v1.1: `https://huggingface.co/datasets/rajpurkar/squad`
- TriviaQA: `https://huggingface.co/datasets/mandarjoshi/trivia_qa`
- NQ-Open: `https://github.com/google-research-datasets/natural-questions/tree/master/nq_open`
- PopQA: `https://github.com/AlexTMallen/adaptive-retrieval`

**Training Setup** All realistic experiments fine-tune the base model Qwen2.5-0.5B (`https://huggingface.co/Qwen/Qwen2.5-0.5B`) with MS-Swift. Only the weights of the selected module are updated and every other parameter stays frozen: QK updates the query and key projections, VO the value and output projections, Attn all four attention projections, MLP the gate, up, and down projections, and Full all attention and MLP projections. Embeddings, normalization layers, and the tied output head stay frozen in every setup. All runs use AdamW with a peak learning rate of $10^{-4}$, 3% linear warm-up followed by cosine decay, weight decay 0.01, gradient clipping at 1.0, an effective batch size of 16, FP32 master weights with BF16 computation, and a loss on the answer tokens only.

**Task Metrics** CRUXEval-O counts a prediction as correct when the predicted output equals the reference under Python-literal equality. SGD uses the official fuzzy joint goal accuracy, averaged over service frames. For Knowledge Conflict, the strict follow rate is the fraction of PopQA conflict passages for which the first output line equals the answer stated in the passage. For knowledge injection, facts learned is the strict closed-book exact match on the injected facts, and SGD skill retained is the fraction of the 17,283 test frames solved before injection that the model still solves completely.

**Knowledge Retention** Retention is measured on the benchmark items that the untouched base model answers correctly under both of two fixed question orders, the source order and the reversed order rotated by one. An item counts as retained if the fine-tuned model still answers it correctly under both orders. The prompt is `Question: {question}\nAnswer:` without a chat template or demonstrations, decoding is greedy in FP32, and PopQA, TriviaQA, and NQ-Open answers are scored on the first output line with each benchmark's original normalization and alias matching. For context following, the PopQA, TriviaQA, and NQ-Open retention scores use a single pass with lenient containment matching, on the items the base model answers correctly under the same prompt (1445, 535, and 156 items, respectively).