NOT FORGOTTEN, NOT RETRIEVED: A SAME-CHECKPOINT DECOMPOSITION OF FAILURE IN MODULAR CONTINUAL RL

Anonymous authors
Paper under double-blind review

ABSTRACT

An old-task performance drop in continual reinforcement learning can mean that a learned capability degraded or that the agent failed to select a capability that remains available. We introduce an architecture-agnostic diagnostic protocol for modular agents whose expert selection can be overridden, and instantiate it in CM-Dreamer, a modular DreamerV3 agent. At the same checkpoint, we compare a *task-aware reference* that forces the expert assigned to the evaluated task with *learned deployment*, in which the agent's router selects or mixes experts without a task label. Their signed difference is the *retrieval gap*; change in task-aware reference performance across training is retention loss. Acquisition is measured separately. In CM-Dreamer, the same final checkpoint on a three-task Meta-World stream (cw3) scores 0.66 under the task-aware reference and 0.32 under learned deployment. Another checkpoint places its largest router weight on the assigned expert on 97% of steps yet has a retrieval gap of 1.00, whereas inaccurate selection can be harmless when several experts solve the same task. Forced experts and fixed mixtures distinguish these cases. When observations reveal task identity, an observation-based selector without task labels matches the reference. When identity-revealing features are masked, a tested router identifies the task only after early actions make correction ineffective. Freezing the reference computation preserves old-task performance but often prevents new-task acquisition; unfreezing the recurrent state model and decoder improves acquisition at a retention cost. Most checkpoints contain only one usable skill because later tasks were often never acquired. The few two-skill checkpoints demonstrate selection failure but are too few to establish its frequency.

1 INTRODUCTION

Continual reinforcement learning requires an agent to learn new tasks while preserving useful behaviour on earlier ones. World-model agents such as DreamerV3 (Hafner et al., 2025a) provide a strong starting point, but deployment on a task stream adds two challenges: avoiding catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999) and deciding which learned behaviour the current situation requires, without a test-time task label (Khetarpal et al., 2022).

Modular agents make the second challenge explicit. We use the following terms throughout. An *expert module* is a task-specific set of parameters; a *slot* is its addressable position; and the *expert bank* is the set of slots stored by a checkpoint. At each step, a *router* maps the interaction history to weights over the slots. These weights define a *route*, which may select one expert or mix several. During evaluation, the evaluator can override the router. The *task-aware reference* forces all weight onto the slot assigned to the known task, whereas *learned deployment* lets the agent's router choose without receiving the task label. The task-aware reference is a controlled diagnostic, not an upper bound on what the expert bank can do.

Consider CM-Dreamer, our modular DreamerV3 agent, trained on the three-task Meta-World stream cw3: `hammer`→`push-wall`→`faucet-close`. After training, we evaluate it again on the first task without providing its label. A lower success rate is commonly reported as forgetting (Pan et al., 2025), yet an expert that can still solve the task may not be selected at deployment. At the

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

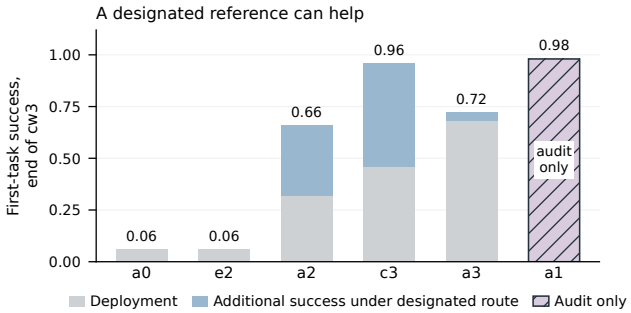


Figure 1: **Task-aware reference and learned deployment measure different failures.** First-task success at the end of cw3. Grey bars show learned deployment; blue segments show the additional success under the task-aware reference. Frozen isolation has no router-based deployment channel, so only its task-aware reference is shown. The gap to 1.0 is not retention loss: it also includes seeds in which the first task was never acquired. Configurations and values are listed in Table 1.

same final checkpoint, CM-Dreamer scores 0.66 under the task-aware reference and 0.32 under learned deployment (Figure 1). Overriding only the route recovers performance without changing any learned parameter. Conversely, a final score cannot establish that the task was acquired in the first place.

We therefore ask three questions in order. **Acquisition:** did the agent learn the task? **Retention:** does the execution path used as a reference still perform as it did when the task was learned? **Retrieval:** how does learned deployment compare with the task-aware reference at the same checkpoint? Acquisition requires measurements during learning. The other two yield a signed decomposition of the drop from earlier task-aware reference performance to final learned deployment: retention loss F plus retrieval gap G (Section 3). Positive G means that deployment underperforms the reference; negative values are possible because the assigned expert need not be the bank's best route. Plasticity is assessed separately through acquisition and additional training interventions; it is not a third additive term in this identity.

The protocol requires controllable routing and a task-aware reference that can be applied consistently across checkpoints; it does not require a particular world-model architecture. For each checkpoint, we evaluate the task-aware reference, learned deployment, every forced single-expert route and several fixed mixtures. We call this set of same-checkpoint evaluations the *routing matrix*. The forced-expert and mixture evaluations test whether a capability is available outside the assigned slot. Separate *selective-unfreezing* experiments resume training with one component group unfrozen to test which components can improve new-task acquisition.

Contributions. First, we introduce an operational protocol that separates retention measured under the task-aware reference from learned deployment and assesses acquisition independently. It requires only overridable expert routing and a reference fixed across checkpoints, so it applies to adapter banks, masks and task heads alike; CM-Dreamer is our instantiation. **Second**, routing interventions show why argmax route accuracy—the fraction of steps on which the assigned slot receives the largest weight—is an insufficient performance diagnostic. Another expert may retain the same skill, and a mixture aimed at the assigned expert can still fail because of its weights or temporal behaviour. On these streams, the cost of routing errors depends on the experts' relative competence and on whether learned deployment reproduces their training-time routing distribution (Sections 4.1–4.2). **Third**, we use the protocol to distinguish deployment from learning limitations. An observation-based selector matches task-aware reference performance when task identity is observable, whereas masking identity-revealing features can make delayed correction ineffective; selective unfreezing can improve acquisition while damaging retention (Sections 4.3–4.4). These findings are tested on short and longer streams with one backbone. The multi-skill examples support the diagnostic distinction, while their small number limits broader claims about when selection fails.

2 Related Work

World models and access to learned behaviour. The Dreamer family (Hafner et al., 2020; 2021; 2025a) and Transformer world models (Chen et al., 2022; Robine et al., 2023; Zhang et al., 2023; Burchi & Timofte, 2025; Hafner et al., 2025b) motivate our testbed. Outside RL, loss of performance need not imply loss of knowledge: Zheng et al. (2025) study lost task alignment, and Trivedi & Melwani (2026) distinguish storage, representation and accessibility using a trained probe. In continual RL, Nijjer (2026) probes a monolithic DreamerV3 agent and attributes degradation primarily to the actor. Our routing interventions compare execution paths without retraining; our separate unfreezing experiments test the consequences of allowing components to change. On our frozen modular agent they reverse that attribution: world-model drift is sufficient for the forgetting in every run, while policy-head drift alone suffices in two of five seeds. The resulting component attribution is specific to our modular setting (Section 4.4).

Task identity and module selection. Regularization (Kirkpatrick et al., 2017; Schwarz et al., 2018) and parameter isolation (Mallya & Lazebnik, 2018; Ben-Iwhiwhu et al., 2023; Huang et al., 2021; Mendez et al., 2022) protect prior learning, but evaluation must also specify how task identity or the corresponding mask is obtained (Wołczyk et al., 2021). Task-agnostic approaches use replay (Rolnick et al., 2019; Kessler et al., 2023) or infer context within a shared representation (Caccia et al., 2023). OWL (Kessler et al., 2022) compares bandit, random, max-Q and oracle head selection on one trained agent. We build on this intervention perspective with an explicit signed reference–deployment decomposition and tests of expert competence and mixtures. Expert Gate (Aljundi et al., 2017) and related approaches (Erden et al., 2025; Römer et al., 2026) supply a label-free selection control; task-free expansion offers another design (Lee et al., 2020).

Experts and plasticity. Low-rank adaptation (LoRA) (Hu et al., 2022) and sparse expert routing (Shazeer et al., 2017; Fedus et al., 2022) make module storage and selection explicit; SD-LoRA (Wu et al., 2025) avoids explicit selection. Work on plasticity examines primacy bias, dormant units and declining effective capacity (Nikishin et al., 2022; Sokar et al., 2023; Lyle et al., 2022; 2023; Dohare et al., 2024; Joudaki et al., 2025). We use component-wise unfreezing to locate an acquisition constraint in our testbed, without claiming to identify its underlying mechanism.

3 A Diagnostic Protocol for Modular Continual RL

We first define the measurements independently of CM-Dreamer, then specify the architecture and streams used to instantiate them.

Setting and routing channels. A task stream $\mathcal{T} = (\tau_1, \ldots, \tau_K)$ shares observation and action spaces and may differ in reward and initial-state distribution. Training identifies task boundaries so that the agent can allocate expert modules. At evaluation, the agent acts from its interaction history H_t without a task label. The evaluator retains labels and can override routing. An *arm* is one agent configuration. As introduced in Section 1, a slot is an addressable expert module and the expert bank is the set of slots held by a checkpoint. A *routing channel* is a rule that maps H_t to expert weights. For each task, the evaluator fixes a task-aware reference channel whose meaning is unchanged across checkpoints. In our implementation it is the one-hot selection of the slot assigned to that task, denoted ta. The agent's own router defines learned deployment, denoted infer. Tables abbreviate these channels as *reference* and *deployment*, respectively.

Retention and retrieval. Let $\theta^{(k)}$ be the checkpoint after task k, and let $R_i^{(k,m)}$ be expected success on task i under routing channel m. Three readouts are needed: reference success when training on task i ends, reference success after later training, and deployment success at that later checkpoint. For $k > i$,

$$F_i^{(k)} \;=\; R_i^{(i,\mathrm{ta})} - R_i^{(k,\mathrm{ta})}, \qquad G_i^{(k)} \;=\; R_i^{(k,\mathrm{ta})} - R_i^{(k,\mathrm{infer})}. \tag{1}$$

Thus $R_i^{(i,\mathrm{ta})} - R_i^{(k,\mathrm{infer})} = F_i^{(k)} + G_i^{(k)}$. F measures change in task-aware reference performance; G is the retrieval gap between that reference and learned deployment at one checkpoint. Routing interventions hold learned parameters, expert bank, environment seeds and initial states fixed. Changes

in actions and subsequent recurrent states are part of the measured effect. Both terms are signed. In particular, $G < 0$ is possible: the task-aware reference is not an upper bound on the bank's competence (Appendix C.3). A small F must be read together with the initial acquisition level; it cannot establish a skill that was never demonstrated.

Acquisition and bank competence. The task-aware reference measurements do not establish whether another slot or mixture can solve the task. We therefore also evaluate forced slots and fixed mixtures. The full set of channels is listed in Table A.1. Acquisition is measured when the task is trained and compared with a single-task model trained from scratch. To investigate limited acquisition, we resume a checkpoint with one component group unfrozen and compare it with its same-seed frozen control. These *selective-unfreezing* experiments require additional training; they assess plasticity separately from Equation 1.

CM-Dreamer and the evaluated arms. CM-Dreamer combines DreamerV3's discrete-latent recurrent state-space model (RSSM) (Hafner et al., 2025a) with per-task LoRA experts on the reward, actor and value heads. A bounded causal Transformer produces a context m_t; a router maps it to expert weights α_t. The context does not write into the RSSM, but can enter the actor and value networks. We call the input builder and this Transformer the *context path*. After the first task, the encoder, RSSM, decoder, continuation head and head base weights are frozen. Earlier expert slots are also frozen. The current slot, context path and router remain trainable in the modular stack.

The main comparisons are frozen isolation, which omits the context path; the modular stack, the full configuration just described, in which the current slot, the context path and the router all keep training; and an anchored arm, which adds retention distillation and retrieval-weighted slot initialization: it constrains old-task behaviour in function space and copies a retrieved old expert into each new slot at initialization. Routing controls use random or task-aware routing during training, and an external gate during deployment. Naive fine-tuning, replay, online elastic weight consolidation (EWC) and Learning without Forgetting (LwF) provide additional controls. Complete training and deployment configurations are in Table A.3. Only frozen isolation has an invariant task-aware reference computation by construction: the other modular arms can change reference performance through their shared context, even with a frozen backbone (Table A.4).

Streams and evaluation. We use Meta-World proprioceptive control (Yu et al., 2019), 10^6 environment steps per task and five seeds per arm unless stated otherwise. In **cw3**, the first three Continual World tasks (Wołczyk et al., 2021) are `hammer`→`push-wall`→`faucet-close`; observations reveal task identity. In **Masked-B2** (hereafter the *masked stream*), `faucet-open`→`faucet-close`, identity-revealing observation dimensions are masked and initial-state ranges aligned. Three held-out masked pairs (window, handle, drawer) and an eight-task learnable subset of Continual World (CW8) test additional task pairs and stream length.

Boundary evaluations use 10 episodes per task and seed. The routing matrix re-evaluates checkpoints at 200 episodes per task and channel. A checkpoint is *capable* at a task if any evaluated channel reaches success 0.5; a *slot-level skill* requires some slot deployed alone to reach that threshold. Argmax route accuracy A_{route} is the fraction of evaluated steps whose largest inferred weight is on the assigned slot. It measures neither the full weight vector nor the consequences of early routing errors.

Experimental controls. Environment steps and update schedules are matched across arms, and storage is matched within three storage regimes (parameter-only, compressed-latent, raw-replay); total parameter counts are not matched. The modular stack has 12.62M parameters, with per-arm counts and storage in Table A.5. Same-seed unfreezing comparisons use homogeneous hardware and deterministic training. The stream-screening criteria and reproducibility details are reported in Appendix A.

4 Experiments

We first distinguish retention under the task-aware reference from learned deployment, then examine why route accuracy does not determine the retrieval gap. We next test when deployment can match reference performance and, finally, which training interventions improve acquisition.

Table 1: cw3 end-of-stream success ($n = 5$ seeds; $\pm$ seed s.d.). The two `hammer` columns compare the task-aware reference with learned deployment on the same checkpoints. Later-task columns use the listed channel; * denotes the task-aware reference. Dashes indicate no evaluated deployment channel, and = denotes a monolithic agent without a module bank. Each seed uses 10 episodes per task. Table B.1 repeats this matrix with each arm's average performance over the three tasks and its storage regime; per-arm parameter and storage budgets are in Table A.5.

		hammer			
configuration	channel	reference	deployment	push-wall	faucet-close
Naive fine-tuning	plain	=	0.06±0.13	0.00	1.00
Replay	plain	=	0.06±0.09	0.00	1.00
Online EWC	plain	=	0.06±0.09	0.06	0.98
Frozen isolation	task-aware*	0.98±0.04	—	0.02	0.12
Modular stack	inferred	0.66	0.32±0.41	0.00	0.22
Anchored arm	inferred	0.72	0.68±0.33	0.00	0.00
Random-trained	random	0.96	0.46±0.42	0.00	0.02
Oracle-trained	task-aware*	0.80±0.45	—	0.02	0.20
Gated arm	gate	0.96	0.96±0.05	0.00	0.32
Single-task reference	task ID	=	1.00	**0.80**	0.80

4.1 Retained reference performance can be lost at deployment

Table 1 separates two questions that a final score conflates. Naive fine-tuning, replay and online EWC end cw3 with first-task success 0.06. Frozen isolation instead retains the first task at 0.98, with $F = 0.00$ within evaluation resolution. Its invariant task-aware reference also retains success 1.00 in all three CW8 seeds through seven task boundaries (Table A.8). This retention does not imply continued learning: the same frozen-isolation runs acquire none of the 21 later-task cells in CW8.

The modular stack illustrates a separate deployment limitation. At the same final cw3 checkpoints, task-aware reference success is 0.66 but learned deployment success is 0.32. The anchored and random-trained arms also differ between reference and deployment. These are within-checkpoint comparisons, not differences between separately trained agents. Reference performance itself is not guaranteed: an unanchored shared context can damage the reference computation, as in the modular-stack seed whose task-aware reference success falls from 0.90 to 0.00 (Table B.5).

Acquisition must be checked before interpreting either result. For `push-wall`, non-isolating arms can acquire the task and subsequently lose it, whereas frozen modular arms fail to acquire it (Appendix C.1). The diagnostic comparisons do not establish an overall performance advantage for modularity; the complete matrix, including average performance and single-task reference ratios, is retained in Table B.1.

4.2 Route accuracy alone does not determine the retrieval gap

We evaluate 59 modular checkpoints from cw3 and Masked-B2 under the routing channels in Table A.1, except the gate, at 200 episodes per cell. Figure 2 shows why argmax accuracy is insufficient: a checkpoint can route correctly on 97% of steps and still have $G = 1.00$, while another has zero argmax accuracy and $G = 0.00$. What separates these two checkpoints is not how often their routers pick the assigned slot. Two things set the cost of wrong routing on these streams: the competence of the experts a route selects, which training fixed, and how far the deployed weight vector and its timing depart from the routing distribution those experts were trained under. Argmax accuracy measures neither. The rest of this section tests the first factor, and Section 4.3 tests the second on both streams: the one whose observations reveal task identity and the one where they do not.

What an alternative slot can do. Forcing each slot in turn measures whether a different route retains the same task skill. Figure 3a reports the resulting spread between the best and worst slots. On the masked stream, the non-assigned slot falls below success 0.5 in all $12/12$ unanchored check-

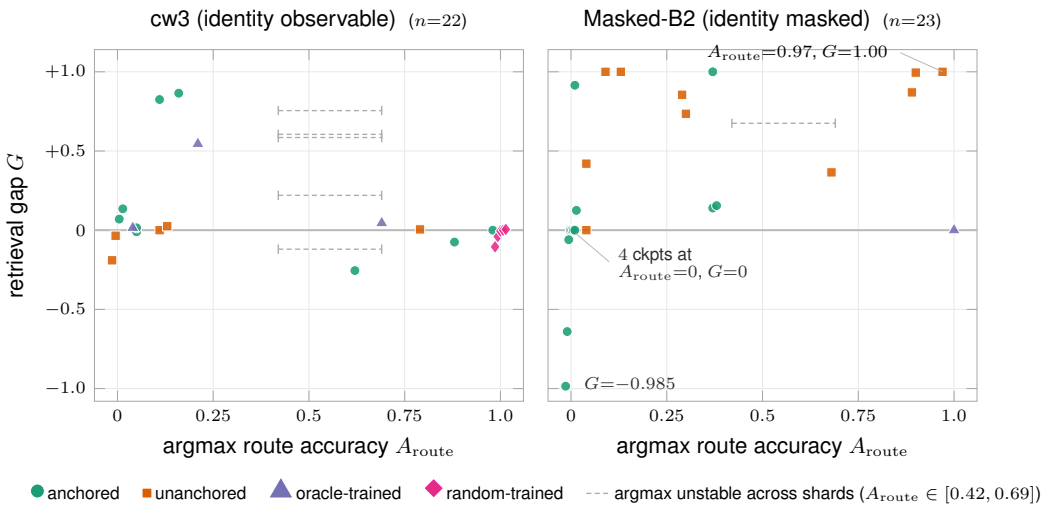


Figure 2: **Argmax accuracy does not determine the retrieval gap.** Each point is a first-task checkpoint from Table A.2; both axes refer to inferred routing. Horizontal axis: argmax route accuracy A_{route}; vertical axis: the signed retrieval gap G of Equation 1. The panels show all 45 capable checkpoints with reported A_{route} (22 cw3, 23 Masked-B2), using the table's rounded values; marker shape and colour give the training family. Six other capable checkpoints lack a single printed accuracy because their argmax flipped between evaluation shards; their gap is known and their pooled accuracy only to lie in 0.42–0.69, so each is drawn as a dashed grey bar spanning that interval (Appendix A). Points tied at $A_{\text{route}} \in \{0, 1\}$ are spread horizontally by at most 0.014. High-accuracy, large-gap and low-accuracy, zero-gap cases both occur, negative gaps occur, and no common correlation law is assumed.

points, whereas it reaches that threshold in $6/14$ anchored checkpoints. In contrast, several cw3 slots retain first-task competence inherited from the backbone. With all slots switched off, the backbone matches the task-aware first-task route within 0.1 on $58/59$ checkpoints, but solves none of the 91 later-task cells. These controls distinguish inherited first-task behaviour from a new skill stored in an adapter. They also explain how inaccurate slot selection can sometimes have little performance cost.

Separating mixture shape, aim and temporal variation. Argmax accuracy ignores the rest of the weight vector. To examine its behavioural consequences, let $\bar{\alpha}$ be the mean inferred weight vector and let $\bar{\alpha}^{\rightarrow}$ swap its largest entry into the assigned slot while preserving the sorted weights. We evaluate four channels on the same checkpoint: these two constant rules, the task-aware one-hot, and the inferred router:

$$G = \underbrace{R^{\text{ta}} - R^{\bar{\alpha}^{\rightarrow}}}_{G_{\text{shape}}} + \underbrace{R^{\bar{\alpha}^{\rightarrow}} - R^{\bar{\alpha}}}_{G_{\text{aim}}} + \underbrace{R^{\bar{\alpha}} - R^{\text{infer}}}_{G_{\text{tvar}}}. \tag{2}$$

The terms compare mixture shape, the aim of the constant mixture, and constant versus time-varying deployment, respectively. They are signed and depend on this intervention path; the temporal term also includes changes in state visitation. Re-aiming measures the constant surrogate, not a direct correction of every decision by the deployed router. If its largest entry is already on the assigned slot, $G_{\text{aim}} = 0$ by construction. Mean-vector estimation and grouped statistics are detailed in Appendix C.2.

Among the 15 evaluated capable checkpoints with $A_{\text{route}} \geq 0.5$ and a defined re-aiming probe, three have $G \geq 0.5$. Their path terms attribute the gaps to mixture shape or temporal variation (Figure 3b; Table C.1). Thus a large gap can remain even when the assigned slot usually has the largest weight.

When the bank holds more than one skill. A two-skill cw3 checkpoint provides a direct counterexample to successful deployment: `faucet-close` success is 0.945 through one slot, 0.455

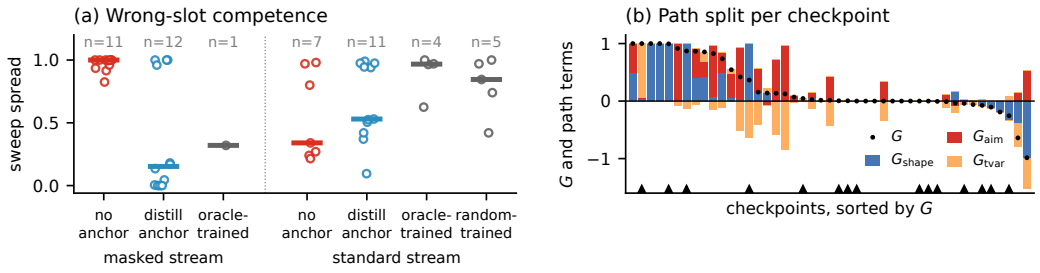


Figure 3: **Expert competence and routing interventions.** First task, 200 episodes per cell. (a) Best-minus-worst success across forced slots, grouped by family and stream; bars are medians. (b) The signed path terms of Equation 2, sorted by G. Triangles mark the high-accuracy group in Table C.1; their mean-vector re-aiming term is zero in this evaluation. Individual checkpoints are reported in Table A.2.

through its assigned slot, and 0.020 through the inferred router. The strongest route is in a different slot from the assigned one (Appendix C.4).

The held-out window stream extends the routing analysis beyond the two main streams. On one `window-open` cell, re-aiming the constant mixture recovers 0.745 against a total gap of 0.750; the other signed terms nearly cancel (Table A.9). Selection can therefore matter when distinct skills are available. Whether deployment can reproduce the routing distribution the experts were trained under is decided by how much task identity the observations carry, and the next section settles it on both streams.

4.3 Recovering reference performance depends on identity and behaviour

Observable identity: a gate recovers the reference. If the second factor of Section 4.2 is the distance between learned-deployment and training-time routing, the gap should close wherever an external selector can reproduce the one-hot distribution the experts trained under. On cw3, an Expert-Gate-style autoencoder selector (Aljundi et al., 2017) can identify the task from observations without a test-time label. The gated arm uses separate runs of the oracle-trained recipe and then deploys through this external gate. On its own checkpoints, gate routing reproduces the task-aware one-hot and matches reference performance on all 15 (seed, task) pairs. First-task success is 0.94 at acquisition and 0.96 at the end of the stream (Table B.4). This comparison establishes deployment of the reference capability, not acquisition of a new one: `push-wall` is still unlearned in all five seeds.

Masked identity: early actions can prevent recovery. The same observation-based gate has first- and second-task route accuracies of 0.49 and 0.66 on Masked-B2. Interaction history contains additional information: episode-held-out linear probes decode task identity from the router's context at 0.77–0.99, yet five of six examined learned routers remain near-constant: over 4800 evaluation steps spanning both tasks, their route accuracy moves at most 4.3% of the way from chance to perfect, and the sixth router's at most 19%. Because that measure spans both tasks while the A_{route} column of Table A.2 reports the first task alone, a near-constant router can still show a high first-task A_{route}. Probe controls and loss variants are reported in Appendix C.5. Decodability over collected trajectories alone does not establish that the deployed router identifies the task early enough to act successfully.

We measure the router's identification time t_{id} as the earliest episode position where positional route accuracy reaches 0.7. Separately, we force a wrong slot for N steps and then switch to the task-aware reference for the remainder of a fixed 250-step episode. The commitment deadline t_{commit} is the largest tested N that retains at least half the task-aware reference success; the grid extends to $N = 128$.

The masked task's measured deadline is 6–8 steps across six checkpoints from four arms. For the oracle-trained router shown in Figure 4a, the identification threshold is reached only in the 33–64-step bin; accuracy subsequently reaches 94%. Its task-aware reference succeeds at 0.600, while

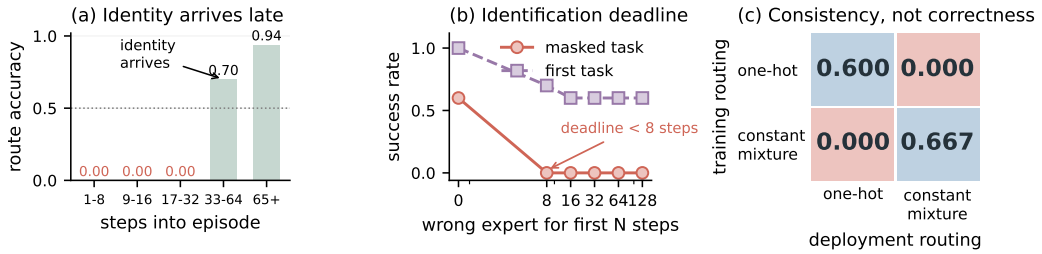


Figure 4: **Late correct routing can fail to recover an episode.** (a) The evaluated oracle-trained router reaches the identification threshold in the 33–64-step bin; this measures the router's accuracy, not the earliest possible availability of identity information. (b) Success after a wrong-slot prefix followed by the task-aware reference. Curves that remain above threshold at $N = 128$ have no observed deadline within the tested range. (c) Crossed training and deployment routing rules, one checkpoint per training rule; each row compares deployment interventions on that checkpoint.

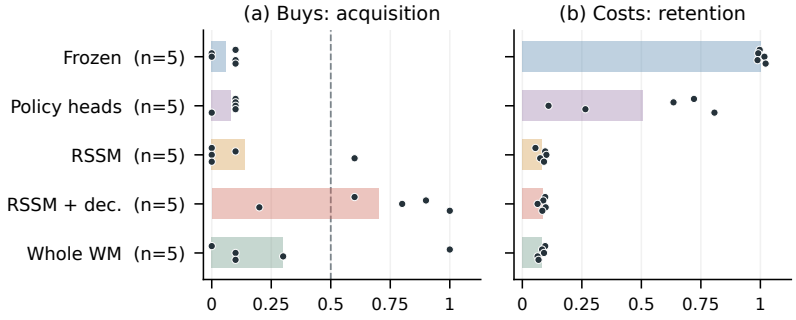


Figure 5: **Single-switch unfreezing, both directions: what unfreezing buys and what it costs.** (Frozen isolation resumed for cw3's second task; dots are runs, n beside each row label; Table B.6.) (a) Acquisition of `push-wall`, peak success over the periodic evaluations. (b) Retention of `hammer` as % of its learned return: every world-model unfreezing keeps 6–10%, unfreezing only the policy heads keeps 63–81% in three seeds and 11–27% in two, the frozen control 99–102%.

learned deployment scores 0.000. Identification therefore arrives well after the 6–8-step deadline measured on this stream, so late correct decisions cannot undo the early behaviour in this example. This is a property of the evaluated router and trajectories, not an information-theoretic lower bound on identification time.

The crossed training/deployment controls in Figure 4c further show that the tested routing rules interact with how the experts were trained (one checkpoint per training rule). Additional prefix and held-out stream experiments show that tolerance of delayed correction also depends on the wrong slot's behaviour (Appendix C.6). These controls support a behaviour-dependent explanation; they do not establish a universal deadline or guarantee that matching marginal routing distributions is enough.

4.4 Selective unfreezing exposes an acquisition–retention trade-off

Routing cannot deploy a capability that was never acquired. On `push-wall`, the architecture trained from scratch reaches 0.80, whereas frozen arms end at no more than 0.02 and acquisition checks confirm their learning failure (Appendix C.1). The limitation extends to CW8: frozen modular arms acquire $3/63$ later-task cells at success ≥ 0.5, compared with $15/21$ for naive fine-tuning (Table A.8). It is task-dependent, however: the frozen oracle-trained arm acquires the second task in $3/3$ drawer seeds and $2/3$ window seeds (Figure A.2).

To locate a trainability constraint, we resume frozen isolation from its first-task checkpoint and unfreeze one component group while training the second task. Figure 5 compares the same-seed interventions; Table B.6 reports every run. On the pre-registered acquisition readout, peak success ≥ 0.5 over periodic evaluations, decoder-containing scopes succeed in $6/13$ runs versus $1/15$ with the

decoder frozen (two-sided Fisher $p = 0.029$). At the boundary evaluation the counts are $3/13$ and $0/15$, respectively. Unfreezing the policy heads alone does not restore acquisition. The comparison implicates adaptation of the RSSM and decoder, while distinguishing transient acquisition from sustained success; it does not isolate a unique underlying mechanism.

The same interventions measure the retention price. Every world-model unfreezing run (18 runs over four scopes) loses 0.86–1.00 of first-task success. Head-only unfreezing loses 0.08–0.22 in three of five seeds and 0.71–0.86 in the other two; the fully frozen control loses none. Thus world-model changes consistently damage task-aware reference performance in this setting, and head changes can also do so. Per-task low-rank residuals on the RSSM and decoder preserve retention but do not restore stable acquisition within the tested placement, ranks, seeds and budget. Their reconstruction improvements and rank sweep are detailed in Appendix B.

5 Discussion and Conclusion

What to report. For a modular continual agent, report acquisition, final task-aware reference performance and final learned deployment together. These distinguish a task that was never learned from degraded reference performance and a retrieval gap. Forced-expert and mixture evaluations then check whether capability exists outside the assigned slot. Route accuracy is useful context, but cannot replace these behavioural measurements. The endpoint differences F and G do not depend on intermediate routing interventions; the finer terms in Equation 2 do.

What transfers beyond CM-Dreamer. The reference–deployment protocol applies when expert routing can be overridden and the same reference can be evaluated across checkpoints. Freezing is not required for the decomposition; it supplies a retention invariant only when the entire task-aware reference computation is unchanged. Our empirical attribution to particular components and routing failures is narrower. CW8 extends the retention and acquisition tests in stream length, while the held-out window cases test routing among multiple available skills.

Scope and limitations. Experiments use one DreamerV3 backbone on Meta-World proprioceptive control, with three-, two- and eight-task streams and three held-out masked pairs. Masking deliberately reduces identity information, an evaluation dimension that remains underexplored (Tang et al., 2026). Window and drawer permit new-task acquisition, but their naive baselines also retain the first task; they therefore do not establish resistance to forgetting. The observed relation between wrong-slot competence and delayed correction is exploratory. Total parameter counts are not matched, hyperparameters are not re-tuned per arm (Mesbahi et al., 2025), and no mask- or ID-conditioned isolation baseline is run, although task-free variants exist (Wortsman et al., 2020). Identifying hidden tasks before harmful actions, while preserving acquisition and retention, remains unresolved.

Conclusion. Task-aware reference retention, learned deployment and new-task acquisition answer different questions, and on these streams the protocol separates them: the same cw3 checkpoint scores 0.66 under the task-aware reference and 0.32 under learned deployment, and a checkpoint whose assigned expert receives the largest router weight on 97% of steps can still lose the task entirely ($G = 1.00$). Accurate argmax routing therefore does not guarantee deployment; what wrong routing costs here tracks the experts' relative competence and the consistency between training-time and deployment-time routing. Where observations carry identity, an observation-based selector matches task-aware reference performance on all 15 (seed, task) pairs; where identity is masked, the routers we examined either stay near-constant or reach their identification threshold only after the 6–8-step commitment deadline. What remains is plasticity, and selective unfreezing places it in the recurrent state model and its decoder ($6/13$ runs acquire against $1/15$ with the decoder frozen), at a retention cost the same protocol makes visible.

AI USE STATEMENT

Large language models were used to polish the writing and to check the internal consistency of reported numbers across sections. They were also used to search the literature on specific research topics. Experimental design, implementation, training, evaluation, analysis and all scientific claims are the authors'.

REPRODUCIBILITY STATEMENT

Training is bitwise deterministic under fixed seeds on homogeneous GPU/driver (Appendix A); every arm's resource vector is in Table A.5, the arms and their one-name conventions in Table A.3, the streams and headroom-screen log in Appendix A, and per-seed values for every main-text number in Tables A.8, B.5, B.6, B.7 and C.4, the per-checkpoint routing matrix (Table A.2) and Figure A.2; wall-clock per run is in Table A.6. Code, configurations, the run manifest and the evaluation scripts that regenerate the tables will be released.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Eseoghene Ben-Iwhiwhu, Saptarshi Nath, Praveen K. Pilly, Soheil Kolouri, and Andrea Soltoggio. Lifelong reinforcement learning with modulating masks. *Transactions on Machine Learning Research (TMLR)*, 2023.

Maxime Burchi and Radu Timofte. Learning transformer-based world models with contrastive predictive coding. In *International Conference on Learning Representations (ICLR)*, 2025.

Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: Gaps and validation of a new baseline. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2023.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

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*, 632:768–774, 2024.

Zeki Doruk Erden, Donia Gasmi, and Boi Faltings. Continual reinforcement learning via autoencoder-driven task and new environment recognition. In *Autonomous Robots and Multirobot Systems (ARMS) Workshop, AAMAS*, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *Nature*, 640:647–653, 2025a.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b.

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

Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Amir Joudaki, Giulia Lanzillotta, Mohammad Samragh Razlighi, Iman Mirzadeh, Keivan Alizadeh, Thomas Hofmann, Mehrdad Farajtabar, and Fartash Faghri. Barriers for learning in an evolving world: Mathematical understanding of loss of plasticity. *arXiv preprint arXiv:2510.00304*, 2025.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7143–7151, 2022.

Samuel Kessler, Mateusz Ostaszewski, Michał Bortkiewicz, Mateusz Żarski, Maciej Wołczyk, Jack Parker-Holder, Stephen J. Roberts, and Piotr Miłoś. The effectiveness of world models for continual reinforcement learning. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2023.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

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, 2017.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision (ECCV)*, 2016.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning (ICML)*, 2023.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

Jorge A. Mendez, Harm van Seijen, and Eric Eaton. Modular lifelong reinforcement learning via neural composition. In *International Conference on Learning Representations (ICLR)*, 2022.

Golnaz Mesbahi, Parham Mohammad Panahi, Olya Mastikhina, Steven Tang, Martha White, and Adam White. Position: Lifetime tuning is incompatible with continual reinforcement learning. In *International Conference on Machine Learning (ICML), Position Paper Track*, 2025.

Gurp Nijjer. The world model remembers, the actor forgets: Dream rehearsal for continual model-based RL. *arXiv preprint arXiv:2607.19749*, 2026.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Chaofan Pan, Xin Yang, Yanhua Li, Wei Wei, Tianrui Li, Bo An, and Jiye Liang. A survey of continual reinforcement learning. *arXiv preprint arXiv:2506.21872*, 2025.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *International Conference on Learning Representations (ICLR)*, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Ralf Römer, Yi Zhang, Yuming Li, and Angela P. Schoellig. CLARE: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *arXiv preprint arXiv:2601.09512*, 2026.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Steven Tang, Xinze Xiong, Anna Hakhverdyan, Andrew Patterson, Jacob Adkins, Jiamin He, Esraa Elelimy, Parham Mohammad Panahi, Martha White, and Adam White. Forager: A lightweight testbed for continual learning with partial observability in RL. *arXiv preprint arXiv:2605.01131*, 2026.

Ayushman Trivedi and Bhavika Melwani. Catastrophic forgetting as accessibility collapse: A three-level framework for knowledge persistence in continual learning. *arXiv preprint arXiv:2606.06032*, 2026.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-LoRA: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations (ICLR)*, 2025. Oral.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *International Conference on Learning Representations (ICLR)*, 2025.

A Reproducibility and Protocol Details

Complete routing-channel inventory. Table A.1 lists the task-aware reference, learned-deployment and diagnostic channels. The core decomposition uses the task-aware (ta) and learned-deployment (infer) channels; the remaining rows probe alternative capabilities or finer routing effects. Forcing the assigned slot reproduces the task-aware channel on all $1500/1500$ evaluated rows in the routing matrix.

Table A.1: Routing channels, grouped as task-aware reference, deployable and diagnostic rows. *label*: whether the evaluator's task label enters the channel; *deployable*: whether an agent could run it without that label; dashes: not a selection rule.

channel	α produced by	label	deployable	used for
task-aware reference (ta)	evaluator's one-hot on task i's slot	yes	no	F, G; *reference* column of Table 1
forced slot j	one-hot on slot j	yes	no	sweep spread; "capable" (bank readout)
learned deployment (infer)	the arm's router, from H_t	no	yes	G; *deployment* column of Table 1
random (rand)	a draw over allocated slots	no	yes	c3; control column of the routing matrix
gate	autoencoder reconstruction error	no	yes	the gated arm c10 (Section 4.3)
backbone only	every slot off (α=0)	—	—	inheritance (Section 4.2)
uniform	equal weight on allocated slots	no	yes	constant reference of G_{sel}
deployment mean $\bar{\alpha}$	step-mean of the inferred α over task i's episodes	yes	no	G_{tvar}, G_{sel}
re-aimed $\bar{\alpha}^{\rightarrow}$	$\bar{\alpha}$ with its largest entry moved to task i's slot	yes	no	G_{aim}, G_{shape}

Training and deployment configurations. The anchored arm adds retrieval-weighted slot initialization and retention distillation, a function-space constraint on the shared context path. The random-trained arm uses random draws during training, while the oracle-trained arm acts on the task-aware one-hot. The latter still builds and trains a router through routing losses and retention distillation; its learned-deployment measurements evaluate that router offline. The gated arm uses separate runs of the oracle-trained recipe with an external autoencoder gate at deployment. Table A.3 gives the complete configuration mapping, including naive fine-tuning, single-task references, replay, online EWC (Schwarz et al., 2018) and monolithic and modular LwF (Li & Hoiem, 2016).

Bitwise reproducibility. With fixed seeds, our training stack is bitwise-deterministic: re-running a seed on the same machine reproduces all logged training metrics exactly (the only deviation is an allocator-level GPU-memory gauge), and machines with identical GPU model and driver reproduce runs bit-for-bit (verified across three machines of a five-machine fleet of identical GPU/driver; 23/23 logged metrics bit-identical). Heterogeneous hardware (different GPU or driver) diverges by 3–7% after the first gradient update; all paired comparisons in this paper were therefore run on homogeneous hardware. The $23/23$ figure covers the logged training metrics on that hardware, not every internal state. Determinism fixes the pairing of single-switch ablations and makes "re-rolling" a failed seed an information-free operation—per-seed failures are properties of the seed, not of run-to-run noise—but it supplies no statistical evidence about differences across training seeds.

Testbed details. The RSSM carries continuous dynamics with recurrent state $h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$, prior $p_\phi(z_t \mid h_t)$ and posterior $q_\phi(z_t \mid h_t, e_t)$; features $f_t = [h_t, z_t]$ drive the reward, actor and value heads exactly as in DreamerV3 (Hafner et al., 2025a). The bounded causal Transformer consumes a sliding window of latents, actions, rewards, continuation flags and a surprise channel and emits the context m_t used by the router; it never conditions the world-model prior and never writes into h_t. On (approximately) shared-dynamics streams the freezing schedule of Section 3 concentrates all forgetting channels into the context path, the router and the task-side residuals, which is what lets the frozen world model's plasticity cost be measured separately (Section 4.4) rather than hidden inside aggregate scores. The router's mixing weights are Top-2 with an episode-level EMA, and the context path (the *shared trunk*) of Section 3 is the token builder together with the Transformer.

Headroom screen log. Candidate streams were screened on naive fine-tuning before unblinding any method results (pre-registered criteria: old task acquired; relative headroom $\Delta F \geq 20\%$ with seed-consistent sign; measurable scratch learning curve on the new task):

candidate stream	naive-baseline statistic	verdict	reason
DMC cartpole pair (s1)	$\Delta F = 0.28\%$	reject	no forgetting to repair
s1 reversed (s1r)	$\Delta F = 83.7\%$, headroom 0.8%	reject	old task never acquired
DMC pair (s2)	$\Delta F = -1.6\%$ / $+2.3\%$ (2 seeds)	reject	sign flips across seeds
sparse variant (s4)	$\Delta F = 13.5\%$ (5 seeds)	reject	below threshold
`faucet` pair (m1)	$\Delta F = 100\%$, but no retrieval gap	converted $\rightarrow$ Masked-B2	masked+aligned by design
cw3 (`hammer`$\rightarrow\ldots$)	$F = 0.68 \pm 0.23$ (5 seeds)	standard regime	headroom confirmed post hoc

The masked regime (Masked-B2) additionally aligns initial-state ranges after an observation-separability audit showed residual identity leakage through initialization ranges (held-out probe residual 0.19 above chance before alignment); MetaWorld task pairs are otherwise fully separable from single observations ($15/15$ pairs at probe accuracy 1.0), which is what motivates a masked pole at all.

The routing matrix, per checkpoint. The full routing matrix over all 59 modular checkpoints (first task of each stream; Section 4.2) is Table A.2. Each of the 59 checkpoints is the end-of-stream checkpoint of a distinct training run (18 arm variants; anchor 27, no-anchor 21, oracle-trained 6, random-trained 5), so checkpoint- and run-level medians coincide and column medians weight every run equally. Runs killed mid-stream by cluster events have no end-of-stream checkpoint and are therefore not in the matrix; the matrix contains every modular run that reached the end of its stream. Denominators: 59 checkpoints (one end-of-stream checkpoint per training run); 51 capable (some channel ≥ 0.5); 150 evaluated cells $=$ 32 cw3 checkpoints $\times$ 3 tasks $+$ 27 Masked-B2 checkpoints $\times$ 2 tasks; 1500 rows $=$ 150 cells $\times$ 10 shards; the re-aimed channel is defined on 45 of the 51 capable checkpoints (52 including seven sub-threshold ones). *random* is the mean of three fixed random Top-k draws; *sweep spread* is best-minus-worst success over forced one-hot slots; $\bar{\alpha}^{\rightarrow}$ and the path terms of Equation 2 are listed per checkpoint, and $G = G_{\text{shape}} + G_{\text{aim}} + G_{\text{tvar}}$ was verified in every row before aggregation. Re-aiming swaps the largest entry of $\bar{\alpha}$ (first index on ties) into task i's slot; a tie for the maximum leaves $\bar{\alpha}$ unchanged, and inactive slots carry zero mass, so no re-aiming targets an unallocated slot. Top-q inclusion is trivially 1 on two-slot streams under Top-2 routing and is not used; every route-accuracy figure in this paper is an argmax accuracy; family assignment follows training-time anchoring (checkpoints of retain-off ablations count as no-anchor). Rows where oracle- or random-trained checkpoints show multi-slot competence with an *unlearned* later task are the second, anchor-free parent of that competence: copy-initialization followed by a later task that never rewrites the slot. This is why the standard stream, whose second task no modular arm in this matrix learns, is delocalized in every family, and separating the two parents here keeps inheritance from being credited to anchoring. On the standard stream an identity-initialized slot trained only on the orthogonal second task keeps the backbone's first-task competence ($10/12$ such slots ≥ 0.8, anchored or not) while a full-copy slot drifts ($2/8$); both families are then immune under mixtures (0.93 vs. 0.67), with $G \approx 0$ for both. Textbook LwF anchors far more weakly than our distillation in both regimes ($7/9$ competent seeds at spread ≥ 0.5).

Table A.2: The routing matrix per checkpoint at 200 episodes per cell (first task). *ta*: task-aware; *inf*: inferred; *rand*: mean of three fixed random draws; *uni*: uniform; $\bar{\alpha}$: deployment-mean constant; $\bar{\alpha}^{\rightarrow}$: re-aimed; *spread*: best minus worst forced one-hot slot; A_{route}: argmax route accuracy. $G = G_{\text{shape}} + G_{\text{aim}} + G_{\text{tvar}}$ holds in every row (verified before aggregation). Path terms and G are printed to three decimals (all rates are multiples of 0.005 at 200 episodes, so the printed identity is exact); channel rates are rounded to two decimals. † not capable (no channel ≥ 0.5).

family	checkpoint	ta	inf	rand	uni	$\bar{\alpha}$	$\bar{\alpha}^{\rightarrow}$	spread	G	G_{shape}	G_{aim}	G_{tvar}	A_{route}
anchor	`a3/cw3_s0`	0.99	1.00	0.98	0.95	0.88	0.99	0.99	-0.010	-0.005	+0.115	-0.120	0.05
anchor	`a3/cw3_s100`	0.99	0.39	0.53	0.53	0.46	—	0.97	+0.605	—	—	—	—
anchor	`a3/cw3_s200`	1.00	0.98	0.99	0.97	0.99	1.00	0.94	+0.015	+0.000	+0.010	+0.005	0.05
anchor	`a3/cw3_s300`	0.53	0.60	0.67	0.63	0.63	0.63	0.51	-0.075	-0.100	+0.000	+0.025	0.88
anchor	`a3/cw3_s400`	0.12	0.37	0.48	0.42	0.45	0.45	0.37	-0.255	-0.330	+0.000	+0.075	0.62
anchor	`a3ao/cw3_s0`†	0.28	0.20	0.24	0.23	0.10	0.24	0.28	+0.080	+0.040	+0.140	-0.100	0.14
anchor	`a3ao/cw3_s100`	1.00	1.00	1.00	1.00	1.00	1.00	0.10	+0.000	+0.000	+0.000	+0.000	0.98
anchor	`a3b2/B2_s0`	1.00	1.00	1.00	1.00	1.00	1.00	0.01	+0.000	+0.000	+0.000	+0.000	0.00
anchor	`a3b2ao/B2_s0`†	0.06	0.03	0.00	0.03	0.04	—	0.06	+0.035	—	—	—	—
anchor	`a3b2aowu/B2_s0`	0.00	0.98	0.64	1.00	0.46	1.00	0.18	-0.985	-1.000	+0.540	-0.525	0.00
anchor	`a3b2ri0/B2_s200`	1.00	1.00	1.00	1.00	0.98	1.00	0.14	+0.000	+0.000	+0.015	-0.015	0.00

family	checkpoint	ta	inf	rand	uni	$\bar{\alpha}$	$\bar{\alpha}^{\rightarrow}$	spread	G	G_{shape}	G_{aim}	G_{tvar}	A_{route}
anchor	a3b2ri0/B2_s300	1.00	0.88	0.44	0.94	0.03	1.00	1.00	+0.125	+0.000	+0.970	-0.845	0.00
anchor	a5b2/B2_s0	1.00	1.00	1.00	1.00	1.00	1.00	0.00	+0.000	+0.000	+0.000	+0.000	0.00
anchor	a5b2/B2_s100	0.94	1.00	1.00	1.00	0.96	0.97	0.17	-0.060	-0.030	+0.010	-0.040	0.00
anchor	a5b2/B2_s200†	0.00	0.01	0.02	0.07	0.00	0.00	0.00	-0.005	+0.000	+0.000	-0.005	0.00
anchor	a5b2/B2_s300	0.10	0.74	0.67	0.81	0.34	0.49	0.04	-0.640	-0.390	+0.150	-0.400	0.00
anchor	a5b2/B2_s400	1.00	1.00	1.00	1.00	1.00	1.00	0.00	+0.000	+0.000	+0.000	+0.000	0.00
anchor	c9/cw3_s0	1.00	0.93	0.88	0.99	0.90	1.00	0.53	+0.070	+0.000	+0.100	-0.030	0.00
anchor	c9/cw3_s100	0.97	0.15	0.23	0.10	0.01	0.90	0.97	+0.825	+0.075	+0.895	-0.145	0.11
anchor	c9/cw3_s200	1.00	0.14	0.24	0.71	0.08	0.60	0.94	+0.865	+0.400	+0.520	-0.055	0.16
anchor	c9/cw3_s300	0.97	0.84	0.87	0.99	0.28	1.00	0.53	+0.135	-0.025	+0.720	-0.560	0.00
anchor	c9/cw3_s400†	0.34	0.04	0.05	0.01	0.00	0.01	0.31	+0.305	+0.335	+0.005	-0.035	0.00
anchor	c9b2/B2_s0	1.00	0.86	0.97	0.92	0.93	0.85	0.00	+0.140	+0.145	-0.075	+0.070	0.37
anchor	c9b2/B2_s100	1.00	0.00	0.00	0.00	0.00	0.00	1.00	+1.000	+1.000	+0.000	+0.000	0.37
anchor	c9b2/B2_s200	1.00	0.84	0.25	0.73	0.44	0.85	0.96	+0.155	+0.145	+0.415	-0.405	0.38
anchor	c9b2/B2_s300	1.00	0.09	0.10	0.78	0.00	1.00	1.00	+0.915	+0.000	+1.000	-0.085	0.01
anchor	c9l01/cw3_s0	0.57	0.69	0.19	0.53	0.51	—	0.42	-0.120	—	—	—	—
no anchor	a2/cw3_s0†	0.49	0.10	0.19	0.10	0.14	0.37	0.48	+0.390	+0.120	+0.230	+0.040	0.01
no anchor	a2/cw3_s100†	0.00	0.07	0.03	0.01	0.02	0.01	0.07	-0.070	-0.015	-0.005	-0.050	0.44
no anchor	a2/cw3_s200	1.00	0.99	1.00	1.00	0.99	0.99	0.24	+0.005	+0.005	+0.000	+0.000	0.79
no anchor	a2/cw3_s300	0.97	0.39	0.33	0.49	0.28	—	0.97	+0.585	—	—	—	—
no anchor	a2/cw3_s400	1.00	0.24	0.47	0.46	0.51	—	0.98	+0.755	—	—	—	—
no anchor	a2b2/B2_s0†	0.00	0.00	0.00	0.00	0.00	0.00	0.00	+0.000	+0.000	+0.000	+0.000	1.00
no anchor	a2b2/B2_s100	1.00	0.00	0.00	0.00	0.00	0.52	0.92	+1.000	+0.485	+0.515	+0.000	0.09
no anchor	a2b2/B2_s200	1.00	0.14	0.15	0.20	0.31	0.58	1.00	+0.855	+0.415	+0.275	+0.165	0.29
no anchor	a3b2rt0/B2_s0	1.00	1.00	0.96	1.00	0.66	1.00	0.94	+0.000	+0.000	+0.340	-0.340	0.04
no anchor	a3b2rt0/B2_s100	1.00	0.00	0.00	0.00	0.95	0.95	1.00	+1.000	+0.050	+0.000	+0.950	0.97
no anchor	a3b2rt0/B2_s300	1.00	0.58	0.22	0.66	0.07	0.97	0.96	+0.420	+0.025	+0.910	-0.515	0.04
no anchor	a3b2rt0/B2_s400	1.00	0.27	0.04	0.21	0.15	0.51	1.00	+0.735	+0.490	+0.355	-0.110	0.30
no anchor	a3b2rt0ao/B2_s0	1.00	0.13	0.01	0.01	0.01	0.01	1.00	+0.870	+0.995	+0.000	-0.125	0.89
no anchor	a3b2rt0ao/B2_s100	1.00	0.64	0.00	0.01	0.01	0.01	0.99	+0.365	+0.995	+0.000	-0.630	0.68
no anchor	a3b2rt0ao/B2_s200	1.00	0.01	0.06	0.04	0.01	0.01	0.82	+0.995	+0.995	+0.000	+0.000	0.90
no anchor	a3b2rt0ao/B2_s300	1.00	0.33	0.05	0.18	0.23	—	1.00	+0.675	—	—	—	—
no anchor	a3b2rt0ao/B2_s400	1.00	0.00	0.00	0.00	0.00	0.01	1.00	+1.000	+0.995	+0.005	+0.000	0.13
no anchor	a3rt0ao/cw3_s100	0.80	0.99	0.98	1.00	0.98	1.00	0.27	-0.190	-0.200	+0.015	-0.005	0.00
no anchor	a3rt0ao/cw3_s200	1.00	1.00	0.97	1.00	1.00	1.00	0.21	+0.000	+0.000	+0.000	+0.000	0.11
no anchor	a3rt0ao/cw3_s300	0.28	0.31	0.43	0.20	0.20	0.10	0.80	-0.035	+0.170	-0.095	-0.110	0.00
no anchor	a3rt0ao/cw3_s400	0.95	0.93	0.78	0.92	0.80	0.89	0.34	+0.025	+0.065	+0.090	-0.130	0.13
oracle-tr.	c6/cw3_s0	0.98	0.77	0.76	0.66	0.74	—	0.96	+0.220	—	—	—	—
oracle-tr.	c6/cw3_s100†	0.01	0.00	0.00	0.00	0.00	0.00	0.01	+0.005	+0.005	+0.000	+0.000	0.00
oracle-tr.	c6/cw3_s200	1.00	0.46	1.00	1.00	0.53	0.94	0.62	+0.545	+0.060	+0.410	+0.075	0.21
oracle-tr.	c6/cw3_s300	1.00	0.98	0.94	0.97	0.56	1.00	1.00	+0.015	+0.000	+0.435	-0.420	0.04
oracle-tr.	c6/cw3_s400	0.97	0.93	0.74	0.80	0.97	0.97	0.97	+0.045	+0.000	+0.000	+0.045	0.69
oracle-tr.	c6b2/B2_s0	1.00	1.00	1.00	1.00	1.00	1.00	0.32	+0.000	+0.000	+0.000	+0.000	1.00
random-tr.	c3/cw3_s0	0.87	0.97	0.91	0.94	0.99	0.99	0.84	-0.105	-0.125	+0.000	+0.020	1.00
random-tr.	c3/cw3_s100	0.90	0.94	0.74	0.82	0.88	0.88	0.42	-0.040	+0.020	+0.000	-0.060	1.00
random-tr.	c3/cw3_s200	0.99	1.00	0.57	0.94	1.00	1.00	0.97	-0.005	-0.005	+0.000	+0.000	1.00
random-tr.	c3/cw3_s300	1.00	1.00	1.00	1.00	1.00	1.00	0.74	+0.000	+0.000	+0.000	+0.000	1.00
random-tr.	c3/cw3_s400	1.00	0.99	0.88	0.88	1.00	1.00	1.00	+0.005	+0.000	+0.000	+0.005	1.00

Pooled-$\bar{\alpha}$ re-evaluation. Table A.2 estimates $\bar{\alpha}$ within each 20-episode shard. Re-evaluating all 59 checkpoints with one $\bar{\alpha}$ per checkpoint, pooled over its 200 episodes and held fixed, reproduces the task-aware and inferred channels bit-for-bit, satisfies the path identity on every row, and changes G_{aim} on the 52 shared cells by a median of 0.000 and at most 0.105 (one cw3 checkpoint). The six capable routers excluded from Table A.2 because their argmax flipped between shards have argmax accuracy 0.42–0.69: at the 0.5 threshold they split three aimed and three mis-aimed ($18/33$ of the 51; $17/34$ at 0.6, $21/30$ at 0.4), the other 45 do not move, and one of the three, with accuracy between 0.5 and 0.6, has its pooled $\bar{\alpha}$ argmax on the wrong slot ($G_{\text{aim}} = 0.075$). Under that split the mis-aimed group's median re-aiming gain is 0.10 (maximum 1.00) and every aimed checkpoint with $G \geq 0.5$ remains shape- or tvar-dominated, so the reading of Section 4.2 is unchanged; the main text keeps the per-shard estimate.

Resource accounting. Table A.5 instantiates the resource vector of Section 3 for the arms of Table 1. Parameter counts are measured by building each arm on the MetaWorld observation space (the count depends on the encoder's first layer, so it is re-measured per benchmark); the frozen arms' trainable count after task 0 is the LoRA bank plus whatever the arm keeps trainable (transformer,

Table A.3: Evaluated arms. "Routing" is the deployment-time source of α; *retain* is function-space distillation of old behaviour (we call arms that carry it **anchored**); *init* is retrieval-weighted initialization of a newborn expert. The *name* column is the one term we use for that arm in the prose—each arm has exactly one.

arm	name used in text	LoRA	context	routing	retain	init	question answered
a0	naive fine-tuning	–	–	–	–	–	how much is forgotten naively
a1	frozen isolation	✓	–	oracle	–	–	is frozen isolation enough
a2	modular stack	✓	✓	inferred	–	–	can the agent find its modules
a3	anchored arm	✓	✓	inferred	✓	✓	full mechanism
c3	random-trained	✓	✓	random	✓	–	does *correct* selection matter
c6	oracle-trained	✓	✓	oracle	✓	✓	value of correct routing in training
c4	single-task reference	single-task scratch, a3 architecture					per-task reference (not a bound)
e1/e2	replay / EWC	replay / online EWC, width-matched					standard families
c9	modular LwF	LwF distillation on the modular stack					distillation family
e3	monolithic LwF	LwF on the monolithic network					distillation control
e4	external gate	✓	✓	AE gate	–	–	external task-agnostic gate
c10	gated arm	✓	✓	AE gate	✓	✓	oracle recipe (separate runs) + gate

Table A.4: Which modules update after task 0, per arm, and whether the task-aware reference computation depends on them. "upd." = trained after task 0; "reference" = on the computation path of the task-aware channel. The retention invariant of Section 4.1 holds where no module on the reference path updates: only for a1. Entries were checked against the code (freeze schedule, actor and value inputs, running statistics).

module	a1 upd.	a2 upd.	a3 / c3 / c6 / c10 upd.	a0 upd.	on reference path
encoder	–	–	–	✓	✓
RSSM (recurrent state model)	–	–	–	✓	✓
decoder	–	–	–	✓	training only
continuation head	–	–	–	✓	✓
reward / actor / value bases	–	–	–	✓	✓
LoRA slot of the current task	✓	✓	✓	n/a	✓(own slot)
LoRA slots of earlier tasks	–	–	–	n/a	✓
context path (shared trunk)	n/a	✓	✓	n/a	✓(actor / value input)
router	n/a	✓	✓[a]	n/a	– (bypassed)
token-builder running statistics	n/a	✓	✓	n/a	✓(via m_t)
actor return-normalization EMA	✓	✓	✓	✓	training only
RMSNorm affine weights (no running statistics)	–	–	–	✓	✓(frozen with their module)

[a] c3 attaches no routing loss (random routing); whether retention distillation still moves its router parameters was not verified.

token builder, router and auxiliary heads for the modular stack; only the LoRA banks for a1). Storage is derived from the run configuration: retention distillation keeps per task a reservoir of 256 posterior start states in fp16 ($256 \times (512{+}2048)$ values $= 1.3$ MB) and a router key, plus—during training only—a snapshot of token builder, transformer and router (≈ 9 MB per task); LwF holds one teacher (the previous model, ≈ 49 MB for the monolithic arm) during training and nothing at deployment; online EWC stores a Fisher diagonal and an anchor ($2P$ floats ≈ 99 MB); shared-quota replay stores $5{\times}10^5$ raw proprioceptive transitions in total (≈ 0.18 KB each, ≈ 90 MB) with per-task quotas of that total; the external gate stores one $\sim$ 700-parameter autoencoder per task ($39 \to 8 \to 39$, 671 parameters), trained at each boundary on 4096 observations reservoir-sampled from the agent's own training-time interaction on that task (no additional interaction, so nothing is debited). Every arm receives 10^6 environment steps per task under the same update schedule and replay ratio, so environment interaction and update count are matched; per-update compute differs with parameter count (9.74 M for the naive arm against 12.62 M for the modular arms); FLOPs were not logged, so Table A.6 reports wall-clock per run together with the number of runs that shared the GPU. Retrieval-weighted initialization consumes context episodes that are debited from the training budget rather than added to it. Top-2 routing makes two of eight slots mathematically active while the vectorized bank is touched in full, so $P_{\text{active}} = P_{\text{total}} - \frac{6}{8}P_{\text{bank}}$ and $P_{\text{touch}} = P_{\text{total}}$ for the modular arms; the difference is at most 0.6 M parameters here.

Table A.5: Resource vector, per arm. P_{total} measured on the MetaWorld observation space; $P_{\text{trainable}}$ = parameters updated after task 0; storage per the text (configuration-derived). Regimes as in Table 1. The single-task reference (c4) trains one full model per task (3×12.62 M).

arm	regime	P_{total}	$P_{\text{trainable}}$ (after task 0)	persistent storage	training-only
a0 naive fine-tuning	—	9.74 M	9.74 M	—	—
e1 replay	R	12.37 M	12.37 M	5×10^5 raw transitions ($\approx$ 90 MB)	—
e2 online EWC	P	12.37 M	12.37 M	Fisher + anchor ($\approx$ 99 MB)	—
e3 monolithic LwF	P	12.37 M	12.37 M	—	one teacher ($\approx$ 49 MB)
a1 frozen isolation	P	10.50 M	0.75 M	LoRA bank (in P_{total})	—
a2 modular stack	C	12.62 M	2.74 M	LoRA bank	—
a3 anchored arm	C	12.62 M	2.74 M	bank + 1.3 MB/task reservoir + key	teacher $\approx$ 9 MB/task
c3 / c6 (routing ablations)	C	12.62 M	2.74 M	as a3	as a3
c9 modular LwF	C	12.62 M	2.74 M	bank	one teacher ($\approx$ 9 MB)
c10 gated arm	C	12.62 M	2.74 M	as a3 + $\sim$ 3 KB/task gate	as a3
c4 single-task reference	—	3×12.62 M	12.62 M/task	—	—

Table A.6: Wall-clock per training run (W&B run time, hours) for the arms of Table 1, with the number of runs that shared the GPU in the same job. Runs were packed two to four per GPU, so GPU-hours per run lie between the wall-clock divided by that number and the wall-clock itself; FLOPs were not logged. cw3 runs are 3×10^6 environment steps (8×10^6 for CW8, not listed); c4 trains one model per task (10^6 steps each).

arm	runs	wall-clock per run (h): median (min–max)	runs per GPU
a0 naive fine-tuning	5	25.0 (16.4–25.0)	2–4
e1 replay	5	31.0 (17.7–31.0)	2–4
e2 online EWC	5	17.4 (17.2–25.0)	2–4
a1 frozen isolation	5	26.0 (17.5–34.0)	2–4
a2 modular stack	5	21.1 (19.4–21.9)	2
a3 anchored arm	5	23.1 (20.6–37.0)	2–3
c3 random-trained	5	34.0 (22.1–37.0)	2–4
c6 oracle-trained	5	22.9 (22.8–30.0)	2–4
c10 gated arm	5	37.0 (11.7–37.0)	1–2
c4 single-task reference (per task)	10	8.5 (8.4–15.7)	2–4

Protocol audit. Table A.7 records which budget axes prior continual-RL evaluations control, on five criteria we hold ourselves to: capacity-matched comparison (D1), quantified storage of old-task information (D2), whether a task ID is required at inference (D3), a selection-correctness control for methods that select modules (D4), and a reported naive-baseline forgetting headroom (D5). Entries are from the cited papers' own reporting; ✓ = controlled/reported, ○ = partial, × = not reported, n/a = not applicable; every entry was checked against the cited paper's text. OWL (Kessler et al., 2022) controls D4 (bandit, random, max-Q and oracle head selection on one trained agent); no prior protocol we found reports D1, D2 and D4 together.

Screening a longer stream (CW8). To test whether our conclusions are limited by stream length, we screened the remaining seven tasks of the Continual World ordering by training the naive baseline on each in isolation under the same per-task budget, three seeds each, with a pre-registered $\geq 2/3$-seeds rule scored on success at the exact budget rather than on return (`stick-pull` reaches a respectable return while never succeeding). Five tasks pass (`push-back`, `handle-press-side`, `window-close`, `push`: $3/3$; `peg-unplug-side`: $2/3$, failing on the first seed we tried—the same seed-bimodal acquisition seen throughout) and two do not (`shelf-place` $1/3$, `stick-pull` $0/3$). A longer stream is therefore feasible for this backbone, but as a *pre-registered learnable subset*, not standard CW10: adding tasks the backbone cannot learn would only dilute every aggregate. We call the eight-task ordering (CW10 minus `shelf-place` and `stick-pull`) CW8. A two-seed pilot at half the per-task budget (5×10^5) failed its pre-registered feasibility criterion (three to four tasks per seed below 0.5)—decisively, because `hammer` is task 0 and cannot suffer sequence interference, yet reached only $0.6/1.0$ against $1.0/1.0$ at the full budget—so the CW8 matrix runs at 10^6 steps per task (naive fine-tuning, frozen isolation, anchored and oracle-trained arms, three seeds each, 8×10^6 steps per run) and is reported per seed in Table A.8. The pilot also showed sequence interference to be task-selective (`push-back`, `push`

Table A.7: Protocol audit of prior continual-RL evaluations against the five budget axes of Section 3. Works: EWC (Kirkpatrick et al., 2017); CLEAR (Rolnick et al., 2019); HyperCRL (Huang et al., 2021); OWL (Kessler et al., 2022); Continual World (CW) (Wołczyk et al., 2021), with PackNet as evaluated there; Modulating masks (Ben-Iwhiwhu et al., 2023); 3RL (Caccia et al., 2023); Continual-Dreamer (Kessler et al., 2023). WM = world model. $\times^{=}$: the method adds no parameters over its own baseline, so capacity is equal by construction, but no capacity-matched comparison is discussed. ∘ in D2: storage reported as sample counts, frames, per-parameter constants or a parameter ratio, not bytes. OWL's D5: an ablation with numbers but no forgetting metric. Continual-Dreamer's D5: forgetting of the FIFO-buffer DreamerV2 baseline. †oracle, random and max-Q head selection compared; random also performs well. ‡heads are selected by the oracle task identity; no random-head control. 3RL's D1: an ablation shows that larger networks do not help, but no parameter-matched comparison is run; its D5: a naive fine-tuning baseline is plotted without a forgetting metric; its D4 is n/a for the method itself, though the same paper compares oracle against entropy-based task-agnostic head selection for its multi-head baselines. *oracle routing is used only as the task-aware reference channel. Our own D1 is ∘: environment steps, update schedule and within-regime storage are matched, total parameter count is not (Table A.5).

work	family	D1 capacity	D2 storage	D3 task ID	D4 selection	D5 headroom
EWC	regularization	$\times^{=}$	∘ (constants)	no (inferred)	n/a	✓
CLEAR	replay	$\times^{=}$	∘ (frames)	no	n/a	✓
HyperCRL	hypernetwork (MBRL)	×	×	yes	n/a	✓
OWL	heads + bandit	×	∘ (counts)	no (bandit)	✓†	∘ (no metric)
Continual World	benchmark (7 methods)	∘	∘ (counts)	yes	×‡	✓
PackNet (in CW)	mask isolation	$\times^{=}$	∘ (1.5×)	yes	×	—
Modulating masks	mask isolation	×	∘	yes	×	∘
3RL	replay + RNN	∘	×	no	n/a	∘
Continual-Dreamer	replay + WM	$\times^{=}$	∘ (counts)	no	n/a	∘ (FIFO)
this paper	modular retrieval + WM	∘ (steps, schedule, storage)	✓	no*	✓ (random)	✓ (gate)

and `peg-unplug-side` acquire far worse in-sequence than from scratch; `faucet-close`, `handle-press-side` and `window-close` are unaffected).

Result. Five of the twelve runs were interrupted by a cluster-wide event mid-stream; each was resumed from its last boundary checkpoint (‡; resume restores agent, optimizer, RNG and module state, and the resumed segment is logged as a separate run), and all twelve are complete. Counts below are over the completed matrix. Three findings. (a) *The plasticity wall is general*: across the frozen modular arms, $3/63$ later-task cells are acquired at ≥ 0.5 (`window-close` in one full-mechanism and one oracle-trained seed, `faucet-close` in a further oracle-trained seed) where naive fine-tuning acquires $15/21$ (Figure A.1); the tasks the pilot flagged as interference-sensitive (`push-back`, `push`, `peg-unplug-side`) are also the ones naive fine-tuning acquires least reliably in sequence, and no frozen arm acquires `push-wall` in any seed.

(b) *Retention replicates and sharpens*: frozen isolation keeps `hammer` at 1.0 in $3/3$ seeds through seven boundaries, while sequential fine-tuning ends the stream retaining only its *last* task (1.0 in $3/3$) and nothing else above 0.3—the eight-task version of the three-family skeleton. (c) *New at this length*: the arms whose shared trunk is trainable and held only by a function-space anchor lose first-task success in one seed each ($0.9 \to 0.4$ anchored, $0.6 \to 0.3$ oracle-trained), whereas the parameter-frozen arm loses none; together with the LwF result below (retains on two tasks, collapses on three) this says distillation-style retention is sequence-length dependent, and parameter freezing is not. With n=3 (five of the twelve runs resumed from a boundary checkpoint after the cluster event, ‡ in Table A.8) we claim these as replications of the three-task structure, not as new effect sizes.

Measurement validation at 200 episodes. Re-evaluating every cw3 second-task boundary checkpoint at 200 episodes (ten disjoint 20-episode shards per cell) reproduces the original 10-episode readouts—14 run-matched pairs, maximum absolute difference 0.23, frozen arms exactly 0 in both—so the resolution upgrade sharpens bounds without moving any conclusion (Figure A.3, right). The same data separates the two routes to `push-wall`'s end-of-stream zero (Figure A.3): the non-isolation arms sit at the bottom right (acquired, then lost), the frozen and modular arms at the origin (never acquired, $0/800$ episodes).

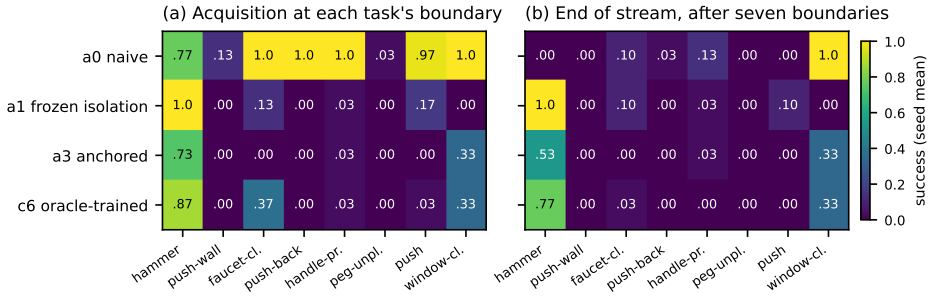


Figure A.1: **Eight tasks, seven boundaries (CW8): only parameter freezing retains through the stream.** Seed-mean success per cell (three seeds; per-seed values in Table A.8). (a) Acquisition at each task's own boundary: naive fine-tuning acquires $15/21$ later-task cells at ≥ 0.5, the frozen modular arms $3/63$—the plasticity wall of Section 4.4. (b) End of stream: naive fine-tuning keeps only its last task, frozen isolation keeps the first task at 1.00 in every seed, and the arms whose trunk is held only by a function-space anchor lose it in one seed each.

Table A.8: CW8 matrix, per seed: *acquisition/end-of-stream* success under each arm's task-aware channel (acquisition = success at the task's own boundary; end = after the last completed boundary). Tasks in stream order. ‡ = run interrupted mid-stream by a cluster event and resumed from its last boundary checkpoint; all twelve runs are complete.

arm / seed	ham	pwall	fclose	pback	hpress	peg	push	wclose
a0 naive fine-tuning s0	0.4/0.0	0.4/0.0	1.0/0.0	1.0/0.0	1.0/0.2	0.0/0.0	0.9/0.0	1.0/1.0
a0 naive fine-tuning s100	1.0/0.0	0.0/0.0	1.0/0.3	1.0/0.1	1.0/0.2	0.0/0.0	1.0/0.0	1.0/1.0
a0 naive fine-tuning s200	0.9/0.0	0.0/0.0	1.0/0.0	1.0/0.0	1.0/0.0	0.1/0.0	1.0/0.0	1.0/1.0
a1 frozen isolation s0	1.0/1.0	0.0/0.0	0.0/0.0	0.0/0.0	0.1/0.1	0.0/0.0	0.1/0.2	0.0/0.0
a1 frozen isolation s100	1.0/1.0	0.0/0.0	0.4/0.3	0.0/0.0	0.0/0.0	0.0/0.0	0.2/0.0	0.0/0.0
a1 frozen isolation s200‡	1.0/1.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.2/0.1	0.0/0.0
a3 anchored arm s0	0.3/0.3	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0
a3 anchored arm s100‡	0.9/0.4	0.0/0.0	0.0/0.0	0.0/0.0	0.1/0.1	0.0/0.0	0.0/0.0	0.0/0.0
a3 anchored arm s200‡	1.0/0.9	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	1.0/1.0
c6 oracle-trained s0	1.0/1.0	0.0/0.0	0.3/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0
c6 oracle-trained s100‡	0.6/0.3	0.0/0.0	0.1/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.1/0.0	1.0/1.0
c6 oracle-trained s200‡	1.0/1.0	0.0/0.0	0.7/0.1	0.0/0.0	0.1/0.0	0.0/0.0	0.0/0.0	0.0/0.0

Re-aiming on the held-out window stream. The three held-out masked streams (Figure A.2) were selected from the observation-separability audit and masked by its minimal masking set (post-mask single-frame probe accuracy ≈ 0.5 at natural resets, so no initial-state alignment is applied); the oracle-trained arm is trained for 2×10^6 steps with three seeds per stream, and in-episode identity probes exceed a same-task null by at most 0.005. The three window-stream checkpoints (oracle-trained arm, both tasks, 200 episodes per cell, pooled $\bar{\alpha}$) give a setting where two competent slots coexist (Table A.9). In the three cells whose $\bar{\alpha}$ argmax is on the wrong slot, re-aiming alone accounts for $G_{\text{aim}} = 0.52$, 0.75 and 0.24 against G_{shape} of -0.07, 0.15 and 0.00; in the one aimed cell with a large gap ($G = 0.50$) the gap is again shape (0.68) with a negative G_{tvar}. These examples show that aim can matter when distinct skills coexist; its cost is smaller when an alternative slot also solves the task.

B Additional Results

Additional acquisition examples. The modular stack peaks at 0.640 and 0.915 on cw3's third task in two of five seeds. These learning-time peaks are distinct from the end-of-stream readouts in Table B.1. The oracle-trained two-skill example and the held-out window and drawer acquisition results are discussed in Sections 4.2 and 4.4.

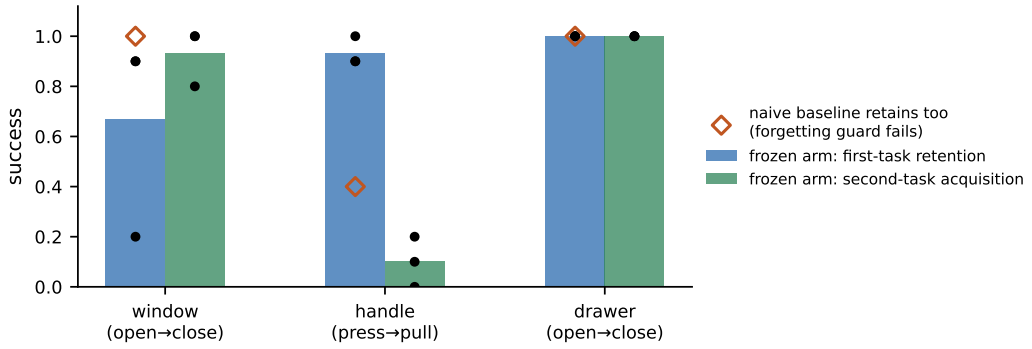


Figure A.2: The three held-out masked streams (three seeds each; diamonds: the naive fine-tuning's end-of-stream first-task success). `window` and `drawer` show a frozen arm acquiring the second task while keeping the first ($2/3$ and $3/3$ seeds), but naive fine-tuning keeps the first task too (diamond at 1.0), so the forgetting guard fails and neither stream can support a retention claim; `handle` passes no acquisition gate at all. Where forgetting pressure is real, a frozen arm acquires while keeping the first task only in some seeds: on the masked pole in $2/5$ seeds—0.535 and 0.640 at 200 episodes, against 0.380 with an upper confidence bound of 0.449 for the best of the rest—against the $3/5$ registered for that test, and on cw3 for its third task in two of five seeds but for its second task in none.

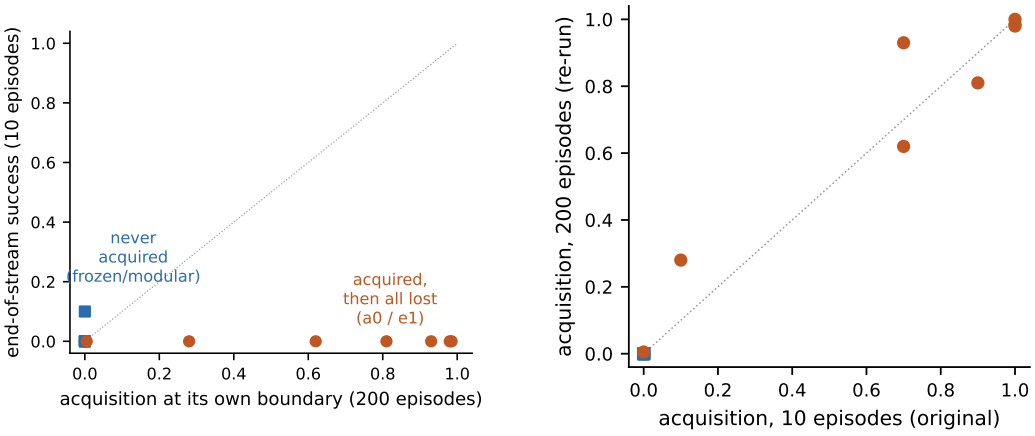


Figure A.3: *Left*: one end-of-stream zero, two routes—`push-wall` acquisition at its own boundary (200 episodes) against end-of-stream success (original 10-episode evaluation; the routing matrix's 200-episode sweep covers only the modular arms, so the uniform 10-episode axis is used for all runs). Circles: non-isolation arms; squares: frozen/modular arms. *Right*: 10- versus 200-episode acquisition, 14 run-matched pairs.

EWC penalty sweep. Online EWC's retention does not depend on its penalty weight over two orders of magnitude on cw3 (Table B.2): first-task success at the end of the stream is 0.00 at $\lambda \in \{10^2, 10^3, 10^4\}$ (n=1 each, seed 0), the last task is learned at 1.00 in every setting (so no setting is "too strong"), and the only cell that moves (`push-wall`) is the seed-bimodal one. We therefore fixed λ=10^4 and ran four further seeds: end-of-stream first-task success $0.00/0.00/0.20/0.10$ (seed 400 never acquired `hammer` and is excluded from the forgetting statistic). Together with the λ=10^4 sweep run (seed 0, 0.00) these are five seeds at one setting, mean 0.06—numerically identical to sequential fine-tuning's 0.06, for which EWC pays $2\times$ parameter-scale storage (Fisher diagonal plus anchor).

Replay: retention at two tasks, collapse at three. Shared-quota replay (one buffer, equal quota per task, raw transitions) retains the first task while the stream has two tasks and loses it when a

Table A.9: Held-out window stream under pooled $\bar{\alpha}$: four channels and the path split of Equation 2, 200 episodes per cell. $^{\times}$: the $\bar{\alpha}$ argmax is on the wrong slot, so the aim cost is measurable; in the other rows $G_{\mathrm{aim}} \equiv 0$ by construction.

checkpoint	task	ta	inf	$\bar{\alpha}$	$\bar{\alpha}^{\rightarrow}$	G	G_{shape}	G_{aim}	G_{tvar}
c6x3/window_s0	window-open$^{\times}$	0.49	0.39	0.04	0.56	+0.105	-0.070	+0.520	-0.345
c6x3/window_s0	window-close	0.57	0.63	0.57	0.57	-0.060	+0.000	+0.000	-0.060
c6x3/window_s100	window-open$^{\times}$	0.94	0.18	0.04	0.79	+0.750	+0.150	+0.745	-0.145
c6x3/window_s100	window-close	1.00	0.50	0.33	0.33	+0.500	+0.675	+0.000	-0.175
c6x3/window_s200	window-open	0.94	0.98	0.98	0.98	-0.040	-0.035	+0.000	-0.005
c6x3/window_s200	window-close$^{\times}$	0.99	0.74	0.75	0.99	+0.245	+0.000	+0.240	+0.005

Table B.1: Main matrix on cw3 (n=5 seeds; $\pm$ seed s.d.; resolution 0.1); read the two hammer columns as a pair. *deployment*: end-of-stream success under the arm's learned-deployment channel (—: no deployment channel evaluated during training); *reference*: the same checkpoint under the task-aware reference (=: no module bank); their difference is G, computed against the listed deployment channel rather than *infer* where the two differ (c3 random, c10 gate). Later-task columns use the arm's channel; AP: mean of the three; AP/ref.: % of the single-task reference, with and without push-wall, which no continual arm ends above 0.06 (Appendix C.1). *plain*: monolithic, no routing; regimes: P parameter-only, C compressed-latent, R raw-replay; *later-task columns and AP are task-aware reference readouts; c10: separate runs of the oracle-trained recipe; c3: one random draw.

arm	regime	channel	hammer (deployment) ↑	hammer (reference) ↑	push-wall ↑	faucet-close ↑	AP ↑	AP/ref. (w/, w/o push-wall)
a0 naive fine-tuning	—	plain	0.06±0.13	=	0.00	1.00	0.353	41/59
e1 replay	R	plain	0.06±0.09	=	0.00	1.00	0.353	41/59
e2 online EWC	P	plain	0.06±0.09	=	0.06	0.98	0.367	42/58
a1 frozen isolation	P	task-aware*	—	0.98±0.04	0.02	0.12	0.373	43/61
a2 modular stack	C	inferred	0.32±0.41	0.66	0.00	0.22	0.180	21/30
a3 anchored arm	C	inferred	0.68±0.33	0.72	0.00	0.00	0.227	26/38
c3 random-trained	C	random	0.46±0.42	0.96	0.00	0.02	0.160	18/27
c6 oracle-trained	C	task-aware*	—	0.80±0.45	0.02	0.20	0.340	39/56
c10 gated arm (oracle recipe + gate)	C	gate	0.96±0.05	0.96	0.00	0.32	0.427	49/71
c4 single-task reference	—	task ID	1.00	=	**0.80**	0.80	0.867	100/100

third arrives. Along the seed-0 trajectory the first task's success under replay is $0.86/0.83/0.84$ at $1.2/1.5/2.0\times10^6$ steps (buffer share 50%) and $0.03/0.02$ at $2.5/3.0\times10^6$ steps (share 37.5%); over five seeds the end-of-stream first-task success is $0.00/0.20/0.00/0.10/0.00$ (mean 0.06) after all five had acquired it (1.00 in four, 0.90 in one). Whether the collapse is a share effect ($1/K$) or a task-pair interference effect is not decided by these runs; either way the replay arm's failure sharpens with the number of tasks (Section 5).

External families on the masked stream. The three-family skeleton of Section 4.1 replicates verbatim on Masked-B2 (Table B.3; three seeds per arm, 2×10^6 steps, no test-time task ID for any arm): the non-isolating families (replay, EWC) learn the new task and lose the old one; LwF-style distillation—whether on the monolithic network (e3) or on the modular stack (c9)—keeps the old task at 1.00 and never learns the new one ($0/5$ and $0/4$). Distillation's retention, however, is *sequence-length dependent*: on cw3 the same monolithic LwF arm ends the three-task stream with first-task success $0.10/0.00/0.90/0.00/0.00$ (median 0.00) while still learning nothing new (push-wall $0/5$, faucet-close $0.20/0.00/0.00/0.00/0.00$), whereas on the two-task Masked-B2 it keeps the first task in $5/5$—so what is unconditional about function-space anchoring of old behavior is the suppression of new-task learning, not the retention (cf. the CW8 reading of Section 5). Three of the five cw3 seeds ($0/100/200$) resume from a task-1 boundary checkpoint after an infrastructure failure and so observe only the third task; the two added seeds ($300/400$) run the stream from scratch and are therefore the only ones in which acquisition and loss of the first task are observed within a single continuous run—both acquire it at 1.00 and end at 0.00.

External gate on the masked stream: confusion rows. Per seed (0/100/200/300/400) the raw-observation autoencoder gate of Section 4.3 routes the first task at accuracy $0.28/0.68/0.46/0.39/0.63$ (mean 0.490) and the second at $0.71/0.65/0.64/0.70/0.62$ (mean 0.663); balanced accuracy 0.576 (sd 0.068; Youden J=0.15)—weakly above chance, and essentially the 0.583 a *supervised* linear probe with true labels reaches on the same masked observations,

Table B.2: Online EWC on cw3, penalty sweep (seed 0, one run per row): success at acquisition (L) and at the end of the stream (F) per task. The retention readout (final0) is 0.00 at every λ; the last task is learned everywhere.

λ	hammer L	hammer F	push-wall L	push-wall F	faucet-close F	AP
10^2	1.00	**0.00**	1.00	0.10	1.00	0.367
10^3	0.80	**0.00**	0.60	0.10	1.00	0.367
10^4	0.80	**0.00**	0.70	0.30	1.00	0.433

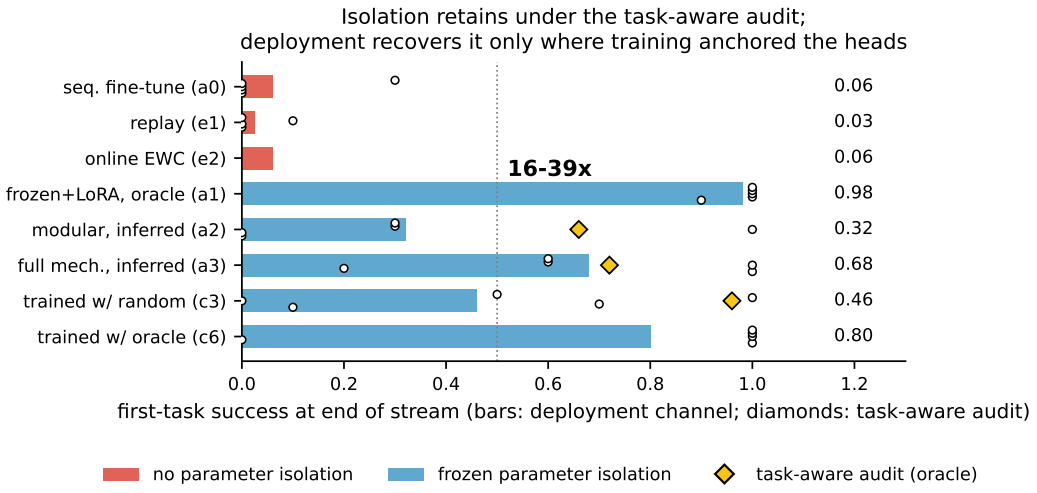


Figure B.1: Per-seed view of the three-family skeleton (the aggregate is Figure 1 and Table 1). First-task success at the end of cw3. Bars/dots: each arm's own deployment channel (means/seeds); diamonds: task-aware reference where it differs. Replay, EWC and fine-tuning are indistinguishable near zero; every isolation bank that acquired the task retains it under the reference (c6's zero seed is an acquisition failure, flagged by the gate), while deployment matches the reference level only for the anchored arm (a3) and oracle-deployed arms. Family gap 16–39×.

so the gate is not under-fit (per-task reconstruction MSE 0.0006–0.0014; the same gate on unmasked `faucet` observations routes at 1.000). The confusion rows expose a bias an accuracy average hides: the newer expert absorbs 58.7% of steps on average (0.71/0.48/0.59/0.65/0.50), and the one seed whose two autoencoders fit equally well (0.0010/0.0010) is the one unbiased seed (0.48)—the bias is an asymmetry of absolute reconstruction quality, inherent to reconstruction-error gating, not an implementation artefact. Never-activated slots are never selected.

The gated arm on cw3, per seed. Table B.4 gives c10's acquisition and end-of-stream success per seed, with the return kept alongside: the gate routes at argmax accuracy 1.000 in all 15 cells, and its channel equals the task-aware channel there cell for cell, so the deployment and reference readings of this arm are one number. First-task success does not degrade over the stream, but the return it keeps is seed-variable (61–119%), so we claim the success level and not the return. Per-seed AP is 0.500/0.333/0.300/0.533/0.467 (mean 0.427). The oracle-trained arm whose recipe it shares has per-seed AP 0.50/0.43/0.40/0.33/0.03; its 0.340 mean is pulled down by the single seed that never acquired `hammer`, and over its four acquisition-successful seeds it averages 0.415. The two arms are separate training runs (the gated arm has no acquisition failure), so the AP difference in Table 1 is seed composition, not the gate; what the gate establishes is that on its own checkpoints deployment equals the task-aware reference. The two arms use the same five seeds but were trained on different hardware, and the gate draws no training-time randomness by construction (reservoir sampling and autoencoder fitting use dedicated generators, verified at the unit level); bit-identity of the two arms' training was not verified end-to-end.

Table B.3: External families on the masked two-task stream (Masked-B2), per seed: first-task (`faucet-open`) success at the end of the stream = retention; second-task (`faucet-close`) success = acquisition. Single deployment channel (no module bank) except c9.

arm	regime	retention (task 0)	acquisition (task 1)
e1 shared-quota replay (K=2)	R	0.00 / 0.60 / 0.00	1.00 / 1.00 / 0.90
e2 online EWC (λ=10^3)	P	0.00 / 0.00 / 0.10	1.00 / 1.00 / 1.00
e3 LwF, monolithic (n=5)	P	**1.00** $\times 5$	0.00 $\times 5$
c9 LwF on the modular stack (n=4)	C	**1.00** $\times 4$	0.00 / 0.00 / 0.00 / 0.20
a0 sequential fine-tune (headroom guard)	—	0.00	learned

Table B.4: The gated arm (c10) on cw3, per seed. *acquisition*: success at the task's own boundary; *end*: at the end of the stream under the arm's deployment channel, which for c10 is the gate—identical to the task-aware channel in all 15 cells. *return kept*: the end-of-stream return as a fraction of the return at acquisition, from the same evaluations; success and return are reported together because a success rate that moves without its return is the signature of clipping a completion condition rather than recovering a capability. `faucet-close` is the last task, so its two columns coincide; `push-wall` sits at the task's return floor throughout.

task	acquisition (seeds 0–400)	end of stream	return kept
`hammer`	0.70 / 1.00 / 1.00 / 1.00 / 1.00	0.90 / 1.00 / 0.90 / 1.00 / 1.00	61–119%
	mean 0.94	mean 0.96	mean 92%
`push-wall`	0.00 / 0.00 / 0.00 / 0.00 / 0.00	0.00 / 0.00 / 0.00 / 0.00 / 0.00	— (at floor)
	mean 0.00	mean 0.00	—
`faucet-close`	0.60 / 0.00 / 0.00 / 0.60 / 0.40	unchanged	— (last task)
	mean 0.32	mean 0.32	—

Where the plasticity wall is: single-switch unfreezing on cw3. Resuming frozen isolation (a1) and the modular stack (a2) from their first-task checkpoints (seed 100) and training only the second task (`push-wall`, 10^6 steps) with one component group unfrozen (Table B.5): the fully frozen runs stay at 0.00; unfreezing RSSM+decoder lets a1 reach 0.90 (peak) / 0.70 (boundary) and unfreezing the whole world model 1.00 (peak); RSSM alone gives 0.10 (a1) and 0.40 (a2, its best cell). Every unfrozen run loses the first task ($1.00 \rightarrow 0.00$ for RSSM and RSSM+decoder; whole world model $\rightarrow 0.40$ after touching 0.00). One seed per cell *as displayed*, though the ordering itself rests on five runs per world-model scope in that ordering (Table B.6); we claim that order (frozen $\approx$ RSSM-only $\ll$ RSSM+decoder $\approx$ whole world model on the primary instrument) and the retention price, not the levels.

The same instrument answers a second question the forgetting literature is currently divided on: *which component's drift does the forgetting*. A fifth switch freezes the world model exactly as the default does and unfreezes only the base weights of the actor and value heads (Table B.5, †). It does not help acquisition—`push-wall` stays at 0.10 peak against its paired control's 0.10, so head plasticity alone buys nothing, which is what the probe was pre-registered to test. What it changes is retention: over five seeds the first task loses $0.08/0.10/0.22/0.71/0.86$ of its success and keeps $81/72/63/27/11\%$ of its return, where *every* world-model unfreezing loses 0.86–1.00 of the success and keeps only 6–10% of the return (eighteen runs). The two sets touch at one seed (0.86 against 0.86), and the same ordering holds on the return axis as on the success axis—we report both because a success rate that moves without its return is the signature of clipping a completion condition rather than recovering a capability. **On this instrument world-model drift is sufficient for the forgetting in every run; policy-head drift alone suffices in two of five seeds**; the probe was pre-registered with this as the outcome that would locate the channel. We claim the ordering rather than the level: what survives is not separable into "the heads are harmless" versus "their adapters bound the drift".

Three mechanism-suggested repairs, and what each rules out. All three target the anchoring mechanism of Section 4.2; each was pre-registered, and each fails in a way that excludes one explanation. *Anchor the old, free the new*: restrict the student's distillation mixture to old slots (so the newborn expert receives no anchoring gradient) and identity-initialize it. Masked stream, five

Table B.5: Single-switch unfreezing for the second cw3 task: `push-wall` success (peak over the periodic 10-episode evaluations during the task / at the separate boundary evaluation; the two evaluation sets are disjoint, the periodic one using the trainer's evaluation environments and the boundary one a separately seeded pool) and `hammer` success at the boundary (retention). "frozen" is the paper's default (everything but the LoRA bank frozen after task 0). One seed per cell as displayed; the per-scope counts of Section 4.4 ($6/13$ vs. $1/15$) rest on the per-seed table (Table B.6). Each block is a within-seed comparison against its own frozen control; the seed-0 block was run separately and is paired with its own control rather than read against the seed-100 rows.

arm	seed	unfrozen components	push-wall peak	push-wall boundary	hammer boundary
a1	100	none (frozen)	0.00	0.00	1.00
a1	100	RSSM	0.10	0.20	0.00
a1	100	RSSM + decoder	**0.90**	0.70	0.00
a1	100	whole world model	**1.00**	0.10	0.40
a1	0	none (frozen)	0.10	0.00	1.00
a1	0	**policy heads only**[†]	0.10	0.00	**0.90**
a2	100	none (frozen)	0.00	0.00	0.00[*]
a2	100	RSSM	0.40	0.40	0.00
a2	100	RSSM + decoder	0.10	0.00	0.10
a2	100	whole world model	0.20	0.00	0.00

[†] *Policy heads only* freezes all three world-model components exactly as "frozen" does—it is bit-for-bit the same configuration on the world-model side—and additionally unfreezes the base weights of the actor and value heads, so the single switch is the policy heads and nothing else. It is the only row here that changes the retention answer: every world-model unfreezing costs the first task almost entirely ($1.00 \to 0.00$–0.40), whereas unfreezing the heads costs 0.10 of success and 28% of return ($4650 \to 3365$) on the seed shown. The row generalizes only partly: across five seeds the heads-only switch loses $0.08/0.10/0.22/0.71/0.86$ of first-task success and keeps $81/72/63/27/11\%$ of the return, against 0.86–1.00 lost and 6–10% kept for the eighteen world-model-unfreezing runs; the all-frozen control keeps 99–102% in five runs (Table B.6). *Read this as an ordering, not a level*: what survives cannot be attributed to the heads being harmless as opposed to their low-rank adapters bounding how far they can drift.

[*] a2 keeps training its shared trunk with no anchor even when the world model is frozen; on this seed its first task is lost under both channels ($0.90 \to 0.00$ oracle)—the no-anchor family's trunk drift of Section 4.2, not a routing failure—so its `hammer` column is that known loss, and only its `push-wall` column bears on the wall.

seeds: new-task success $0.00/0.00/0.70/0.00/0.30$ with first-task retention 1.00 in all seeds—two seeds pass the pre-registered criterion where three were required—and the instrument that would show the surgery acting, the mass the student router assigns to the newborn slot on old-task material, reads 0.000–0.009 from the start of the task: there was nothing to free. Standard stream, three seeds: third-task success $0.00/0.00/0.20$ (one seed never acquired `hammer`), the wall being the frozen world model's (Table B.5). *Per-task low-rank world-model adaptation*: give the recurrent state model (including the block-diagonal recurrent gates, via a per-block low-rank residual that preserves the block structure) and the decoder one low-rank residual per task, freeze both base weights after the first task, and take the deployment identity from the external gate (the only identity source that does not pass through the latent, so the routing loop stays broken). Resumed from the same first-task checkpoint as the single-switch dissection, same seeds, same 10^6 steps on the second task, ranks 8 and 32, two seeds each:

arm	2nd-task peak	1st-task end / min	state recon. (start→end)
per-task residual, rank 8	0.00	**1.00 / 1.00**	$0.057 \to 0.049$
per-task residual, rank 32	0.10	**1.00 / 1.00**	$0.065 \to$ **0.022**
fully frozen (reference)	0.00	1.00 / 1.00	$2.332 \to 2.502$
unfrozen both (reference)	**0.90**	0.00 / 0.00	$0.014 \to 0.009$

Only the active slot's residual moves (norm 47.9; every other slot exactly zero) and the world-model losses are attached, so the mechanism is live; the reconstruction column shows the residual recovers most of the frozen-to-unfrozen fit gap without reaching the level the arms that learn attain (offline audit below). *Uniform prefix*: deploy a uniform expert mixture for the first N steps and

Table B.6: Single-switch unfreezing on cw3, one run per (seed, scope) cell (a1 resumed from its own first-task checkpoint per seed; second task `push-wall`, 10^6 steps). Columns: `push-wall` peak over the periodic 10-episode evaluations (the pre-registered acquisition readout; ≥ 0.5 counts as acquired), at the boundary evaluation, and in the last quarter of the task window (Q4); `hammer` forgetting F (curve readout, learned minus final), success at the final boundary evaluation, and return kept (final/learned). Two same-machine reruns that reproduced their originals bit for bit are merged. Decoder-frozen scopes (none, policy heads, RSSM): $1/15$ acquired; decoder-unfrozen scopes: $6/13$, the RSSM+decoder+policy-heads scope having been run on 3 of the five seeds.

seed	unfrozen scope	pw peak	pw boundary	pw end (Q4)	hammer F	hammer boundary	return kept
0	none (frozen)	0.10	0.00	0.00	0.00	1.00	100%
0	policy heads only	0.10	0.00	0.00	0.10	0.90	72%
0	RSSM	0.00	0.00	0.00	0.86	0.00	6%
0	RSSM + decoder	0.60	0.20	0.10	1.00	0.00	10%
0	whole world model	0.00	0.00	0.00	1.00	0.00	10%
0	RSSM + decoder + policy heads	1.00	1.00	0.97	0.98	0.00	10%
100	none (frozen)	0.00	0.00	0.00	0.00	1.00	99%
100	policy heads only	0.10	0.00	0.00	0.22	1.00	64%
100	RSSM	0.10	0.20	0.02	1.00	0.00	10%
100	RSSM + decoder	0.90	0.70	0.29	1.00	0.00	9%
100	whole world model	1.00	0.10	0.45	0.98	0.40	8%
200	none (frozen)	0.00	0.00	0.00	0.00	1.00	102%
200	policy heads only	0.10	0.00	0.00	0.86	0.00	11%
200	RSSM	0.00	0.00	0.00	1.00	0.00	10%
200	RSSM + decoder	0.80	0.00	0.19	0.98	0.00	6%
200	whole world model	0.10	0.00	0.00	1.00	0.00	9%
200	RSSM + decoder + policy heads	0.20	0.10	0.02	0.92	0.10	10%
300	none (frozen)	0.10	0.00	0.01	0.00	1.00	99%
300	policy heads only	0.10	0.00	0.01	0.71	0.60	26%
300	RSSM	0.60	0.30	0.22	0.94	0.30	8%
300	RSSM + decoder	0.20	0.00	0.01	0.99	0.00	10%
300	whole world model	0.30	0.10	0.04	0.96	0.00	6%
300	RSSM + decoder + policy heads	0.10	0.00	0.00	1.00	0.00	10%
400	none (frozen)	0.10	0.00	0.00	0.00	1.00	102%
400	policy heads only	0.00	0.00	0.00	0.08	0.90	81%
400	RSSM	0.00	0.00	0.00	0.98	0.00	9%
400	RSSM + decoder	1.00	1.00	0.53	0.97	0.00	8%
400	whole world model	0.10	0.00	0.00	1.00	0.00	7%

commit afterwards, hoping to defer the identification deadline. On the masked pole a uniform prefix of eight steps already zeroes the one decidable second-task cell, exactly like the wrong expert, and only pushes the *first* task's deadline out—because the frozen backbone *is* the first-task policy (Section 4.2), so every mixture that contains it commits to the first task. A uniform prefix is not behaviourally neutral.

What freezing buys, priced by unfreezing. This is the full accounting behind the closing paragraph of Section 4.4. To price freezing, we let the anchored stack keep training its RSSM through the stream—same seeds and hardware as its frozen twins, one switch, with and without a world-model distillation term (four seeds per arm). Unfrozen, first-task retention collapses in every seed ($1.00 \to 0.00$, $1.00 \to 0.30$, $0.50 \to 0.20$, $0.30 \to 0.00$ against the frozen same-seed runs); the distillation term pins the world model's behavior (post-boundary KL $259 \to 1.1$) yet buys none of it back (0.00 in all four seeds). But the third task, which no frozen anchored run ever learns ($0/5$), is learned in $3/8$ unfrozen runs (two of four without distillation—one under the oracle channel, one only under the learned mixture, 0.20 vs. 1.00—and one of four with it, under both channels), and the two runs whose routers did not collapse (usage entropy ≈ 1.0) are two of these three: the plasticity wall of Section 4.4 is the frozen backbone's, as diagnosed, and freezing is a traded quantity—retention bought with acquisition—rather than a free guarantee. Unfreezing is not the fix, but it locates the wall; the single-switch dissection of Section 4.4 then narrows it to the recurrent state model plus decoder, and prices that pair identically (first-task retention $1.00 \to 0.00$). Per-task adaptation of exactly those two components, with retention kept by freezing their base weights, is the one repair this diagnosis licenses—so we built and pre-registered it, and it fails. Adding a per-task low-rank residual to the recurrent state model (including its block-diagonal gates) and the

decoder, with both base weights frozen, retains the first task *perfectly* (1.00 at the end and 1.00 at every evaluation inside the second task, against 0.00 for unfreezing) and learns the second task not at all (0.00 at rank 8, 0.10 at rank 32, two seeds each, against 0.90 for unfreezing the same two components). The residual is active, but these runs do not isolate a capacity constraint: only the current task's slot moves (its residual norm is 47.9 while every other slot stays exactly zero) and the world-model losses are attached, so the residual recovers 97.7% of the frozen-to-unfrozen reconstruction gap. It does not close it. An offline audit that scores all six arms of the dissection with one decoder call on matched trajectories—the training metric (symlog reconstruction negative log-likelihood) and raw normalized state error side by side—puts the residual $12\times$ and $283\times$ from the arm that learns, respectively (Table B.7); and across those six arms reconstruction quality *rank-orders* learning without an inversion, the two arms that learn holding the two lowest errors and a $6\times$ gap separating them from the four that do not. So we cannot say fit quality is beside the point: on this instrument it tracks learning without exception. We also cannot promote that into a cause. Across these arms, fit level and permission to rewrite the backbone are confounded by construction—every arm that reaches the low fit does so *by* being allowed to rewrite it—so six arms and one boundary can refute "fit is irrelevant" but cannot establish "fit is the binding constraint", and we do not claim it is. What we can say is that the wall is not a frozen world model resisting adaptation: the residual moves it most of the way, and still lands on the wrong side of the fit level every learning arm attains. A rank sweep to 128—sixteen times the head adapters—is now complete and says something more specific than "capacity is irrelevant". End-of-stream second-task success stays at exactly 0.00 at every rank ($8/32/64/128$) in every seed (four seeds at ranks 8 and 32, three at 64 and 128), and first-task retention ends at 1.00 in every run, so the verdict does not move. In the pilot subset (two seeds at ranks 8 and 32, one at 64 and 128) *transient* competence during training rose monotonically with rank: over 100 periodic evaluations of ten episodes each, the number that score non-zero was 0, 1–2, 3, 5 at ranks $8/32/64/128$, and the best single evaluation 0.00, 0.10, 0.10, 0.30. With the completed seed set the rise does not survive: the best single evaluation per seed is $0.2/0.0/0.0/0.2$ at rank 8, $0.1/0.1/0.1/0.1$ at 32, $0.1/0.1/0.4$ at 64 and $0.0/0.3/0.1$ at 128, so the best transient evaluation is 0.40 at rank 64 in one seed and the ordering by rank is not seed-stable. Reconstruction, by contrast, does not order with rank at all and flips between seeds ($0.007/0.032/0.049/0.034$ at seed 200; 0.046 at rank 8 versus 0.016 at rank 32 at seed 100). So added capacity does buy the policy something, and what it buys does not survive to the end of the stream—which is the same shape as the retention/plasticity trade rather than an escape from it. The sweep was pre-registered to confirm the negative if success stayed ≤ 0.2 and to overturn it at ≥ 0.5; the best transient evaluation, 0.40 at rank 64, falls between the two branches, so on the pre-registered criterion the sweep is undecided. We report both readouts: the end-of-stream verdict (0.00 at every rank) and the transient one (0.40 at rank 64). The pilot's pre-registration named the training peak as its readout (thresholds 0.5 and 0.3); the extension to ranks 64 and 128 was pre-registered with thresholds 0.2 and 0.5 without naming a readout, so on the peak the sweep is undecided and on the end-of-stream readout it is a confirmed negative at every rank. Within the tested adapter placement, ranks, seeds and interaction budget, low-rank adaptation did not restore stable acquisition; these runs do not separate insufficient model accuracy from optimization and representation constraints, and a frozen base W_0 with a trained residual BA does not by itself fix the effective map $W_0 + BA$, so no claim about "permission to reorganize" follows from the parameterization. In this design retention and plasticity are two faces of one observed cost: parameter isolation buys perfect retention and no plasticity (Section 4.4).

Inheritance across seven boundaries (eight-task stream). Forced-slot sweeps with the backbone alone ($\alpha = 0$) on the end-of-stream checkpoints of the eight-task run (entry criterion: first-task success ≥ 0.8, leaving three frozen-arm and one anchored-trunk checkpoint; two others fall below and are excluded). First-task success by forced slot, in slot order $1 \ldots 7$, with oracle/backbone-only in front: frozen arm seed 0 $1.00/1.00 \mid 0.10, 0.20, 0.80, 0.90, 1.00, 1.00, 1.00$; seed 100 $1.00/1.00 \mid 1.00, 0.00, 1.00, 0.30, 0.10, 1.00, 0.90$; seed 200 $1.00/1.00 \mid 1.00, 0.00, 1.00, 1.00, 1.00, 1.00, 1.00$; anchored arm seed 0 $1.00/1.00 \mid 0.40, 0.90, 1.00, 0.80, 0.60, 0.00, 0.40$. The backbone keeps the first task exactly in every case; whether a wrong slot inherits it depends on that slot's adapter ($14/21$ within 0.1 of the backbone for the frozen arm, $2/7$ for the anchored arm), and the third task's adapter destroys the first task in all three frozen-arm seeds although it never solved its own task ($0.00/0.30/0.00$). Later-task rows read ≈ 0 under oracle and backbone alike, consistent with Table A.8.

Table B.7: Offline reconstruction audit of the six single-switch arms, scored with one decoder call on the posterior state of matched trajectories (the same condition under which the training metric is computed). **NLL** is the training objective itself (symlog reconstruction negative log-likelihood); **NMSE** is raw per-dimension squared error normalized by a variance shared across arms. Ratios are to the arm that learns the second task. The two metrics disagree by $23\times$ on how far the low-rank residual sits from that arm, which is why the choice of fit metric is not innocent; both nonetheless rank-order learning without an inversion. Scoring each arm on trajectories from its *own* policy rather than the shared set changes the residual's NLL by $1.37\times$, so the ordering is not an artifact of on-policy data.

arm	NLL	×	NMSE	×	learns task 2
unfreeze world model	0.0032	0.6	0.178	1.3	yes
unfreeze RSSM + decoder	0.0057	1.0	0.134	1.0	yes (0.90)
per-task residual, rank 8	0.036	6.4	13.8	103	no (0.00)
per-task residual, rank 32	0.069	12.2	37.9	283	no (0.10)
unfreeze RSSM only	0.197	34.8	1.80×10^{3}	1.3×10^{4}	no
fully frozen	2.78	492	8.10×10^{4}	6.1×10^{5}	no (0.00)

C Extended Results for Section 4

This appendix carries the per-cell and per-seed material behind the conclusions of Section 4, organized by the subsection each supports.

C.1 One end-of-stream zero, two routes (Table 1)

No arm ends cw3 having solved `push-wall`, and the two arm families reach that same 0.00 by opposite routes. The non-isolating arms acquire it at its own boundary—0.449 (fine-tuning) and 0.591 (replay) at 200 evaluation episodes per boundary, 0.675 (EWC, 10 episodes)—and then lose all of it; no frozen or modular arm acquires it at all ($0/800$ episodes over the four re-evaluated runs, 0.00–0.02 at 10 episodes for the rest; Figure A.3). Trained on it alone the same architecture reaches 0.80 ($4/5$ seeds), so neither zero is the task being impossible. The learnability gate was scored on *from-scratch* training, which `push-wall` passes; in sequence it survives only as a task that is learned and immediately lost, or never learned—a mismatch between the condition a task is admitted under and the condition it is used under, which also costs CW8 one further task (`peg-unplug-side`, acquired by no arm including the baseline; Table A.8). Table B.1 therefore gives `push-wall` its own column and reports AP/ref. both with and without it; the second ratio is what it would be on a stream where every task is in-sequence learnable, and the 8–22 point difference between the two is the price of including a task none of these methods reach. The aggregate also hides that this is retention failure in one family and the plasticity wall in the other. For c3 the deployment channel in Table B.1 is a *single* random draw made at deployment time, whereas the *random* column of the per-checkpoint routing matrix (Appendix A) averages three fixed draws and therefore reads higher (0.82 at 200 episodes); the two are different quantities.

What distinguishes freezing from the rest is length, not degree: replay holds the first task at 0.83–0.86 at two tasks and 0.03 at three, LwF in $5/5$ seeds at two and at a median 0.00 at three, and the anchored-trunk arms lose it in one CW8 seed each ($0.9 \to 0.4$, $0.6 \to 0.3$). None of this is free, and none of it is general. The same freezing leaves 0 of 21 later CW8 cells acquired at ≥ 0.5 (Table A.8), so later slots are covered by the architectural argument and by no measurement—there is no second competence to retain. Section 4.4 prices this retention by unfreezing.

C.2 Additional routing diagnostics

Mean-vector estimation and the path terms. Equation 2 uses the step-mean of inferred weights within each 20-episode shard. Re-aiming swaps its largest entry into the assigned slot without changing the sorted mass profile; ties and inactive slots are handled as described in Appendix A. Argmax accuracy is a step-wise statistic, whereas re-aiming intervenes on the mean vector. These are distinct measurements. In the 45 capable checkpoints with the per-shard re-aiming probe, the

Table C.1: Path terms by route-accuracy group. The 45 *re-aimable* checkpoints (Section 4.2; not the 45 one-skill banks) split at route accuracy $A_{\text{route}} = 0.5$ (unchanged at 0.4 and 0.6); medians over checkpoints, first task, 200 episodes per cell. The aimed group's three gaps ≥ 0.5, all on the masked stream, are 1.00 (G_{tvar} 0.95), 0.995 (all shape) and 0.87 (shape 0.995, G_{tvar} -0.125), against a group median of 0.000. G_{aim} is measured against the deployment-mean constant $\bar{\alpha}$, not against the deployed router.

group	n	median A_{route}	median G (IQR)	$G \geq 0.5$ / ≥ 0.3	median G_{aim}	max G_{aim}	median G_{sel}
aimed ($A_{\text{route}} \geq 0.5$)	15	0.97	0.000 $[-0.04, 0.36]$	3 / 4	$\equiv 0$	$\equiv 0$	0.005
mis-aimed ($A_{\text{route}} < 0.5$)	30	0.04	0.048 $[0.00, 0.73]$	9 / 10	0.13	1.00	0.000

high-accuracy group happens to have zero re-aiming terms. This is not a general implication of high step-wise accuracy; the pooled evaluation in Appendix A includes an exception among the previously excluded checkpoints.

A constant-reference comparison. The auxiliary constant-reference gain compares learned deployment with the better of two constant rules, uniform and $\bar{\alpha}$:

$$G_{\text{sel}} = R^{(k,\text{const}^*)} - R^{(k,\text{infer})}.$$

Where the better reference is the router's own mean, this comparison does not test re-aiming. Table C.1 preserves the grouped results; the full per-checkpoint values are in Table A.2.

In the lower-accuracy group, re-aiming the constant $\bar{\alpha}$ recovers a median of 0.13, while the median constant-reference gain is 0.00. However, the constant mean performs worse than learned deployment on 19 of the 30 checkpoints. Relative to the deployed router, the same re-aiming is therefore worth a median of 0.010. On thirteen checkpoints, G_{aim} exceeds the total gap because other path terms are negative. These comparisons illustrate why the terms must be read as signed differences along a specified path.

Additional competence comparisons. On the masked stream, unanchored versus anchored families have median forced-slot spread 1.000 versus 0.153, and median G 0.855 versus 0.00. An earlier same-checkpoint probe of one anchored cw3 seed reports task-aware, inferred and random success $1.00/1.00/0.97$, despite route accuracy 0.072; in the same probe, random routing costs the oracle-trained arm 0.17 (one seed). These are the earlier small-evaluation probe results, not the 200-episode matrix used for Figure 2.

C.3 The task-aware reference is not an upper bound (Section 3)

$G < 0$—the inferred soft mixture beating the task-aware one-hot—is not hypothetical, and its frequency bounds how the one-hot channel should be read. Across the routing matrix $G < 0$ in 28 of 150 evaluated cells (19%; 12 of the 51 cells entering the aggregate of Section 4.2), reaching -0.985. Write $G_{\text{dist}} = G - G_{\text{sel}}$ for the part of the gap the constant-reference gain of Appendix C.2 does not cover. A possible explanation is mismatch between training and reference mixtures: heads trained under soft mixtures are off-distribution when forced to a one-hot, and it can dominate—in one trunk-isolation checkpoint the inferred router scores 0.85 where the task-aware one-hot scores 0.65, with the slot convention verified (forcing the assigned slot reproduces the task-aware channel exactly). Medians are unaffected (the aggregate median G is $+0.025$), but any single-checkpoint comparison against the task-aware channel must allow for the sign.

C.4 Routing matrix: tail, inheritance and constant deployment (Section 4.2)

Tail of the per-checkpoint split. Over the 51 capable checkpoints (the re-aimed probe of Section 4.2 covers 45 of them plus seven sub-threshold checkpoints, 52 in all) the median constant-reference gain G_{sel} is 0.000 and the mean 0.052; 10% exceed 0.2, one reaches 0.95, and of the sixteen checkpoints with $G \geq 0.5$ four are dominated by it—so even the coarse metric shows a tail where selection binds, and the path split of Equation 2 should be read per checkpoint there: twelve of the sixteen enter the re-aimed probe (three aimed, nine mis-aimed; Table A.2 lists their path terms) and the other four are the shard-flip exclusions. Regrouped by checkpoint, the constant-reference

gain has median 0.005 over the 15 aimed and 0.000 over the 30 mis-aimed capable checkpoints, with one aimed checkpoint at 0.95 (the step-to-step-variation case of Section 4.2). The column that read as an exception at ten episodes (oracle-trained, $G_{\text{sel}} = 0.10$ against $G_{\text{dist}} = 0.05$; Appendix C.3) is none at two hundred: $G = 0.045$ (Table A.2). The column medians and the tail are reported at 200 episodes because the two are not distinguishable at the ten-episode resolution: on that scale all seven column medians round to 0.00, and only the 200-episode sweep separates "zero" from "one episode in forty".

Where the constant-reference aggregate comes from. The median of 0.000 reported above is taken over the first task of each checkpoint. Splitting those cells by whether the same checkpoint also holds a slot-level skill at a later task—some slot, deployed alone, reaching 0.5 on it, with the first task held the same way—shows that 45 of the 51 come from banks holding a single skill, where the question the term asks does not arise: with only one competent path, and that path largely the backbone (base-only 0.955–1.000 on those cells), first-task statistics alone do not measure selection among distinct task skills. Six checkpoints of the matrix hold two skills; on their later-task cells the term is no longer centred at zero (Table C.2), and the three whose intervals exclude zero reach 0.130, 0.390 and 0.580. To enlarge this stratum we evaluated, with the same instrument and a criterion fixed before the evaluation, four further masked-stream checkpoints of the oracle-trained arm trained at $1.5\times$ the per-task budget: two hold two skills at 200 episodes (later-task forced-slot success 0.800 and 0.990) and two do not (0.040, 0.000) and serve as controls. The controls reproduce the one-skill reading (G_{sel} 0.030 and 0.000 on their first task); the two new two-skill cells read -0.005 and 0.055. Pooled, two of eight two-skill later-task cells exceed 0.2 against five of forty-five one-skill cells, which the pre-registered criterion classes as undecided (binomial p=0.22 against the one-skill rate; the registered thresholds were $\geq 30\%$ with $p < 0.05$ to claim that selection binds where a second skill exists, and $\leq 15\%$ to retain the aggregate reading). We therefore report the interval and not a direction: eight cells cannot establish how often selection binds, only that the aggregate does not speak to it.

Table C.2: Constant-reference gain G_{sel} by whether the checkpoint holds a second skill. The aggregate row is the 51 capable first-task cells of Section 4.2 (capable: any channel ≥ 0.5); a checkpoint *holds a skill* at a task if some slot, deployed alone, reaches 0.5 on it at 200 episodes, and holds two if that is true of its first task and of a later task. Intervals are shard-level bootstrap. Upper block: the 59-checkpoint routing matrix. Lower block: a pre-registered extension of four masked-stream checkpoints of the oracle-trained arm trained at $1.5\times$ the per-task budget, evaluated with the same instrument; eligibility for the two-skill rows was set by this 200-episode evaluation, not by the boundary readouts, and the pooled row combines both blocks.

cells	n	median	mean	$\lvert G_{\text{sel}}\rvert > 0.2$	CI excl. 0	range
first tasks (the aggregate)	51	$+0.000$	$+0.052$	6/51	24/51	$[-0.630, +0.950]$
one-skill banks	45	$+0.000$	$+0.051$	5/45	22/45	$[-0.630, +0.950]$
two-skill banks	6	$+0.015$	$+0.062$	1/6	2/6	$[-0.025, +0.265]$
later tasks, two-skill banks	6	$+0.065$	$+0.179$	2/6	3/6	$[-0.020, +0.580]$
extension: first tasks, one-skill banks (control)	2	$+0.015$	$+0.015$	0/2	1/2	$[+0.000, +0.030]$
first tasks, two-skill banks	2	-0.015	-0.015	0/2	2/2	$[-0.140, +0.110]$
later tasks, two-skill banks	2	$+0.025$	$+0.025$	0/2	0/2	$[-0.005, +0.055]$
later tasks, two-skill banks, pooled	8	$+0.028$	$+0.141$	2/8	3/8	$[-0.020, +0.580]$

One frozen checkpoint with two competent slots. One cw3 checkpoint (oracle-trained, seed 0) ends the stream competent at two tasks under forced one-hot deployment: `hammer` at 0.985 and `faucet-close` at 0.945, the latter stable across all ten shards (0.80–1.00) at a return of 4229 against a task floor near 400. The backbone alone scores 0.000 on `faucet-close`, so that competence is stored in an adapter rather than inherited; `hammer`, by contrast, is backbone competence (0.995 with every slot off), which is what freezing preserves. Two features make this the sharpest single instance of the retrieval gap. The competence sits in the slot allocated to `push-wall`, a task this run never learned, while the slot allocated to `faucet-close` reaches 0.455; and the deployed router delivers 0.020 of it, against 0.600 for a uniform constant, so this is the largest constant-reference gain in the matrix ($G_{\text{sel}} = 0.580$, bootstrap 95% CI $[0.525, 0.635]$) and its distribution term is negative (-0.145). The two instruments agree on the cell they share: the task-aware boundary readout gives 0.50 at ten episodes where the 200-episode sweep of the assigned slot gives 0.455.

They also agree on which seed: this is the only one of the arm's five in which the third task passes 0.5 under either readout ($0.5/0.1/0.0/0.3/0.1$ at the boundary). Read at the task-aware reference, bank and deployment levels of Section 3, the assigned slot gives 0.455, the bank 0.945 and deployment 0.020: retained, though not in place; available; not retrieved. It is the case in which the task-aware reference understates the bank, and the reason capability is scored over every channel. We report it as an existence proof rather than a rate: what it establishes is that a frozen arm *can* end a three-task stream holding two acquired competences, and that where it does, selection is what fails.

Two parents of multi-slot competence. The *anchor-free* parent is copy-initialization followed by a later task that never rewrites the slot; this is why the standard stream, whose second task no modular arm in this matrix learns, is delocalized in every family, and why cw3 is a degenerate control for the anchoring rule of Section 4.2: no modular arm in this matrix learns the second task, so nothing overwrites the first task's competence. The positive evidence for the rule is the masked stream (n=11 versus 12 checkpoints); cw3 is consistent with it. The per-checkpoint routing matrix (Appendix A) and the eight-task inheritance sweep at the end of Appendix B separate the two parents so that inheritance is not credited to anchoring.

Routing correctness and learnability are separate axes. On cw3 the seed with the most accurate router never learned the third task, and the most confused router learned it best. Mis-mixtures are benign on this stream—a constant-mixture deployment of the oracle-trained arm loses almost nothing (0.90 versus 1.00)—so *time-varying* wrong routing is worse than constantly wrong routing. Consistency between the training-time and deployment-time routing distributions organizes much of what we see on both streams, but it is a regime-scoped observation rather than a law: it says nothing about the aim cost once experts are genuinely distinct (the external gate is formally consistent yet routes at $0.49/0.66$ under masking). In four (checkpoint, task) cells no forced one-hot exceeds 0.2 while a mixture scores 0.55–1.0; there "pick the right expert" is not the right frame. There the bank readout is the mixture itself, which no one-hot channel, assigned or best, reads.

C.5 Different losses, same collapse, different near-constant routers (Section 4.3)

Retention distillation typically yields a near-constant router under which the new task is starved of gradient (0.000; old task kept at 1.000; where the router points can drift late in training—approximate input-independence is the robust invariant). In one of four retain-on configurations probed later, a seed whose router escaped the collapse *did* learn the new task under full distillation pressure, so the result is consistent with collapse-induced gradient starvation and does not isolate the distillation weight as its cause. An auxiliary task-identity cross-entropy drives the near-constant router to the *newest* slot: the CE is perfectly satisfied (loss ≈ 0, training-time identity accuracy 1.000) by a degenerate router, and the overpowered anchor destroys first-task retention as well ($1.00 \to 0.00$, $2/5$ seeds, seed-bimodal). Textbook LwF preserves retention but leaves new-task success at ≈ 0 ($0.00/0.00/0.00/0.20$, four seeds). The external families reproduce their standard-stream skeleton here verbatim (Table B.3): replay and EWC learn the second task (0.9–1.0) and lose the first (mean 0.20 and 0.03), while monolithic LwF keeps the first (1.00, $3/3$) and never learns the second (0.00, $3/3$)—distillation's acquisition block is a property neither of our modular stack nor of the standard stream. Under masked identity the learned router converges, with that single exception, to near-constant routing; the loss mixture only chooses where the router points. When the router has collapsed to its own mean, the constant-reference gain reduces to $\max\{R^{\mathrm{uniform}} - R^{\mathrm{infer}}, 0\}$ and is zero whenever the uniform reference does no better; it is then uninformative about aim, which G_{aim} of Equation 2 measures directly. The informative content here is the collapse, the aim cost and the deadline; the constant-reference split earns its keep on the standard stream, where the router is demonstrably *not* near-constant (argmax accuracy 0.866) and still buys nothing over a constant.

C.6 The identification deadline, per cell (Section 4.3)

Six checkpoints, four arms. Figure 4a shows route accuracy by episode position for the oracle-trained arm: identity arrives in the 33–64 bin, and steady-state routing is then 94% correct. Running the wrong expert for the first N steps and then routing perfectly for the remainder of the fixed 250-step episode (500 environment steps at action repeat 2; N is deducted from, not added to, the

Table C.3: Identity decodability by input and split. Linear (ℓ_2-regularised logistic) probes predicting the task from the router's context m, from in-episode observations and from reset observations; both splits come from one re-evaluation at 2000 steps per task: the step split holds out random steps, the episode split whole episodes (32 per task); under the step split the same six checkpoints decode at 0.91–1.00; null = the same probe on a random within-task split of one task's data into two pseudo-classes (the finite-sample chance floor); real = probe − null. E16: the six checkpoints of Section 4.3, whose route accuracies there come from an earlier run of the same probe; B2-ext: six further Masked-B2 checkpoints.

checkpoint	set	context m, step split		context m, episode split			in-episode obs., episode		reset obs., episode	
		probe	null	probe	null	real	probe	null	probe	null
a2b2 s100	E16	1.00	0.50	0.98	0.56	+0.42	0.71	0.49	0.54	0.47
a2b2 s200	E16	0.91	0.49	0.77	0.42	+0.35	0.58	0.46	0.66	0.43
a3b2ri0 s200	E16	0.98	0.50	0.98	0.51	+0.46	0.70	0.51	0.66	0.43
a3b2ri0 s300	E16	0.98	0.50	0.97	0.54	+0.43	0.87	0.38	0.59	0.55
a3b2rt0 s0	E16	0.99	0.50	0.98	0.43	+0.55	0.72	0.34	0.66	0.65
a3b2rt0 s100	E16	0.99	0.50	0.99	0.58	+0.41	0.94	0.47	0.54	0.47
a3b2 s0	B2-ext	0.98	0.50	0.97	0.40	+0.58	0.52	0.34	0.66	0.65
a3b2ao s0	B2-ext	0.86	0.49	0.81	0.43	+0.38	0.44	0.39	0.66	0.65
a3b2aowu s0	B2-ext	0.99	0.50	0.98	0.41	+0.57	0.74	0.33	0.66	0.65
a3b2rt0ao s0	B2-ext	0.87	0.49	0.77	0.45	+0.32	0.56	0.45	0.66	0.65
a3b2rt0ao s100	B2-ext	0.93	0.49	0.84	0.50	+0.35	0.53	0.39	0.54	0.47
c6b2 s0	B2-ext	0.99	0.51	0.95	0.41	+0.54	0.63	0.40	0.66	0.65

horizon), the masked task collapses once N passes a small threshold. Over six checkpoints spanning four architecturally distinct arms on a grid refined to unit steps, the commitment deadline t_{commit} is 6–8 steps in every one (one cell at 6, five at 8, each read as the last N before the first drop below half; one curve returns above half at a single later grid point), and by $N{=}16$ all six have fallen below half their task-aware success, most of them to 0.00. Four arms that differ in routing, anchoring and trunk isolation agree on the deadline to within one grid point. The oracle-trained arm commits before identity is available; the deadline alone does not settle deployment elsewhere, since in one trunk-isolation checkpoint of this set the deployed router scores 0.85 against 0.65 for the task-aware channel (Appendix C.3). t_{id} is measured under the behaviour the agent actually runs, so it is a joint property of stream and policy; a policy that acted to disambiguate could in principle identify sooner, which is a training-side change rather than a deployment-time rule.

Uniform prefix, per cell. Holding the commitment time fixed and replacing the wrong expert with a uniform-α prefix (a tested alternative, not a behaviourally neutral one), the pre-registered criterion (success at $N \geq 48$ clearing half the task-aware channel) splits the six cells. In two (a7b2, a3b2rt0ao) the uniform prefix survives the whole grid (no deadline observed up to $N{=}128$, against 8 with the wrong expert), clearing the bar by a wide margin—a7b2 holds 0.95 at $N{=}128$ against a bar of 0.325. There the cost was the wrong expert's actions rather than commitment time, and a wait-then-commit rule recovers deployment. In a third (a2b2) the deadline moves from 8 to 16 but the criterion fails at $N{\geq}48$. In two of the three oracle-trained cells the uniform prefix dies at the same point as the wrong one (6, 8); in the remaining one the deadline moves later but the criterion still fails at $N{\geq}48$. Of the two prefixes tested, neither helps the oracle-trained arm and one helps two of the three other arms; this bounds the tested prefixes, not every deployment-time rule. On the masked pole itself a uniform prefix of eight steps zeroes the one decidable second-task cell exactly like the wrong expert and only pushes the *first* task's deadline out, because the frozen backbone *is* the first-task policy (Section 4.2), so every mixture that contains it commits to the first task.

Two controls. Within the identical protocol, the same checkpoint's *first* task degrades gracefully instead (1.00 at $N{=}0$, 0.70 at 8, then a 0.60 plateau through $N{=}128$), and a repair-arm checkpoint's first task survives to $N{=}32$: one grid, deadlines ranging from under 8 to 32–64 to none. Across three held-out masked streams plus the masked pole (14 checkpoint cells: eight and six, the latter including the four seed-paired repair checkpoints) tolerance of delayed correction tracks whether the wrong expert is competent at the task (13 of 13 decidable cells agree; Table C.4); where training delocalized competence across slots, running the wrong expert for 128 steps costs nothing within the tested range.

Table C.4: Delayed correction against wrong-slot competence, per checkpoint cell. *wrong slot*: success on the task with the whole episode forced to the other expert slot; N: success when the wrong expert acts for the first N steps of the fixed 250-step episode and routing is correct thereafter; t_{commit}: the largest N at which success is at least half the task-aware success (*none*: no deadline within the tested range). The criterion predicts a deadline when the wrong slot's success is zero and none when it is above zero; it agrees on 13 of 13 decidable cells when the single 0.05 cell is counted as zero and on 12 of 13 when it is counted as above zero. One cell is left undecided because the two readings of t_{commit} (largest qualifying N; last N before the first drop below half) give opposite verdicts. †The two readings disagree at 10 episodes (*none* vs. 16) and agree at 20 episodes on all three environment seeds of a re-evaluation (no deadline); the other two re-evaluated cells (`drawer_s0`, `drawer_s100`) keep their verdicts. `c6x3`: the oracle-trained arm on the three held-out masked streams of Figure A.2, second task; `B2`: the masked pole, first task (`a3b2rt0`: full-copy slot initialization; `a3b2rt0ao`: identity initialization, the repair arm of Appendix C.7), where two cells were evaluated on a grid that also contains N=48 (`a3b2rt0ao/B2_s0` 0.40; `a3b2rt0ao/B2_s100` 0.65).

checkpoint	task	ep.	oracle	wrong slot	N=0	8	16	32	64	128	t_{commit}	criterion
wrong slot at 0*: deadline predicted*												
`c6x3/drawer_s0`	`drawer-close`	10	1.00	0.00	1.00	0.00	0.00	0.00	0.10	0.00	0	agrees
`c6x3/handle_s0`	`handle-pull`	10	0.10	0.00	0.10	0.10	0.10	0.00	0.20	0.00	64	agrees
`c6x3/handle_s100`	`handle-pull`	10	0.20	0.00	0.20	0.30	0.40	0.30	0.30	0.00	64	agrees
`a3b2rt0ao/B2_s0`	`faucet-open`	20	1.00	0.00	1.00	1.00	1.00	0.85	0.00	0.00	32	agrees
`a3b2rt0ao/B2_s100`	`faucet-open`	20	1.00	0.00	1.00	0.65	0.90	0.75	0.20	0.20	48	agrees
`a3b2rt0/B2_s400`	`faucet-open`	20	1.00	0.00	1.00	1.00	1.00	1.00	0.80	0.20	64	agrees
`a3b2rt0ao/B2_s400`	`faucet-open`	20	1.00	0.00	1.00	1.00	0.30	0.00	0.00	0.00	8	agrees
wrong slot above 0*: no deadline predicted*												
`c6x3/drawer_s100`	`drawer-close`	10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	none	agrees
`c6x3/drawer_s200`	`drawer-close`	10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	none	agrees
`c6x3/window_s0`	`window-close`	10	0.60	0.30	0.60	0.90	1.00	1.00	0.90	0.90	none	agrees
`c6x3/window_s100`	`window-close`	10	1.00	0.30	1.00	1.00	0.80	0.40	0.60	0.80	none†	agrees
`c6x3/window_s200`	`window-close`	10	1.00	0.30	1.00	1.00	1.00	0.90	1.00	0.70	none	agrees
wrong slot at 0.05 *(one success in 20 episodes)*												
`a3b2rt0/B2_s300`	`faucet-open`	20	1.00	0.05	1.00	1.00	0.60	0.15	0.15	0.05	16	agrees as 0, not as >0
undecided: the two readings of t_{commit} *give opposite verdicts*												
`a3b2rt0ao/B2_s300`	`faucet-open`	20	1.00	0.00	1.00	1.00	0.90	0.15	0.95	1.00	none / 16	undecided

Consistency grid. Figure 4c crosses two training-time routing distributions (one checkpoint each) with two deployment rules: both consistent cells work ($0.600/0.667$) and both inconsistent cells score 0.000, including a constant-mixture deployment of an oracle-trained agent. A 94%-correct router scores 0.000; a constant router scores 0.667.

C.7 A SINGLE-SWITCH REPAIR WHERE IDENTITY IS MASKED, AND ITS CW3 REPLICATION

Retrieval-weighted initialization copies a working old expert's full LoRA pair (A, B) into the newborn slot; the newborn is then a functioning replica, symmetry never breaks, and the new task is never learned (0.000 under both routing channels, despite a healthy-looking router). Copying *only* A and keeping B zero—restoring the LoRA convention that a new expert is born as an exact identity—improves new-task success in three of four seed-paired comparisons on the masked stream (differences $+0.70$, $+0.10$, $+0.60$; one tie at 0.00), holding everything else fixed, including keeping retention distillation off in both arms (the two damages are conjunctive, so keeping distillation off isolates the initialization factor). Which channel carries the rescue is seed-dependent, and probed offline the largest rescue is *mixture-only*: neither forced one-hot slot solves the new task ($0.00/0.00$) while the learned mixture does (0.65). We claim the sign (n=5 arms, four paired). The same intervention, pre-registered as a replication on cw3, shows no benefit there: on acquisition-gated seeds the baseline arm learns the third task in $2/3$ seeds (mean 0.50), the identity-initialized arm in $0/3$ (Fisher $p \approx 0.4$). The two regimes fail for different reasons—under masking the newborn expert is a working replica, the disease the fix targets; on cw3 the third task is blocked by the frozen backbone's plasticity wall (the checkpoint with the most accurate router, 0.87, still never learns it)—so what breaks, and therefore what fixes it, is a property of the stream rather than of the architecture alone.