RESET OR REUSE? UNDERSTANDING HOW OPTIMIZER MEMORY SHAPES CONTINUAL MIXTURE-OF-EXPERTS LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual fine-tuning of mixture-of-experts (MoE) models, a paradigm that sequentially adapts a pre-trained MoE model to a stream of tasks, has become a popular method in lifelong model deployment because it acquires new capabilities without retraining from scratch. However, standard sequential fine-tuning carries optimizer state across task boundaries and updates the full parameter space, so cross-task interference from inherited optimization history and from heterogeneous routed experts and always-active modules is neither measured nor controlled, while existing methods instead estimate parameter importance at substantial storage cost. To address this, we investigate the plasticity–forgetting dynamics of continual MoE fine-tuning through boundary-state interventions and role-structured updates, finding that interference degrades both what a new task acquires and what earlier tasks retain, which motivates separating what to discard from what to keep. Accordingly, we propose MoRE, a structured optimizer-update scheme for *moment reset and reuse* that clears the optimizer's fast state at task boundaries, reuses accumulated moment history to adapt the effective learning rate, and allocates subsequent updates across granularities, from role-level gates to per-coordinate modulation. On a six-task sequence from the TRACE continual learning benchmark, MoRE averages 49.53 on OLMoE-Base, versus 45.63 for sequential fine-tuning (SeqFT) and 48.39 for the parameter-regularization baseline EWC; on Qwen, it reaches 54.90, versus 51.98 for SeqFT and 54.60 for EWC; it requires no Fisher estimation pass, and reduces storage for retained coordinate history by $3\times$ relative to EWC's Fisher-plus-anchor state at equal coverage.

1 INTRODUCTION

Large pre-trained language models are widely used in natural language processing, and mixture-of-experts (MoE) architectures increase their capacity through sparsely activated experts (Shazeer et al., 2017; Muennighoff et al., 2024). Continual learning (CL) extends these models to successive tasks and evolving data without retraining from scratch (Wang et al., 2024; 2023). In this work, we study continual fine-tuning of MoE language models, repeatedly training the same expert pool through successive deployments to acquire new capabilities, while retaining those from earlier tasks.

However, standard sequential fine-tuning carries optimizer state across tasks and updates the full parameter space without distinguishing parameter roles. The sparse activation of MoE architectures alone does not yet prevent interference: successive tasks reuse routed experts alongside always-active attention, embeddings, and, in some architectures, shared experts (Muennighoff et al., 2024; Dai et al., 2024; Li et al., 2024). Existing methods address complementary aspects: EWC protects important parameters (Kirkpatrick et al., 2017), MoFO selects updates using momentum (Chen et al., 2024), and ContinualAdam resets task-local moments while reusing past second moments (Kang & Lee, 2024). These mechanisms do not jointly determine which optimizer history to retain and where to allocate updates across heterogeneous MoE roles. This raises our central question: *Which history should be reset or reused, and at what granularity should it control subsequent updates?*

To address this, we first measure task acquisition and subsequent retention under boundary-state interventions, seeking to improve both learning new tasks and preserving earlier capabilities. These

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.

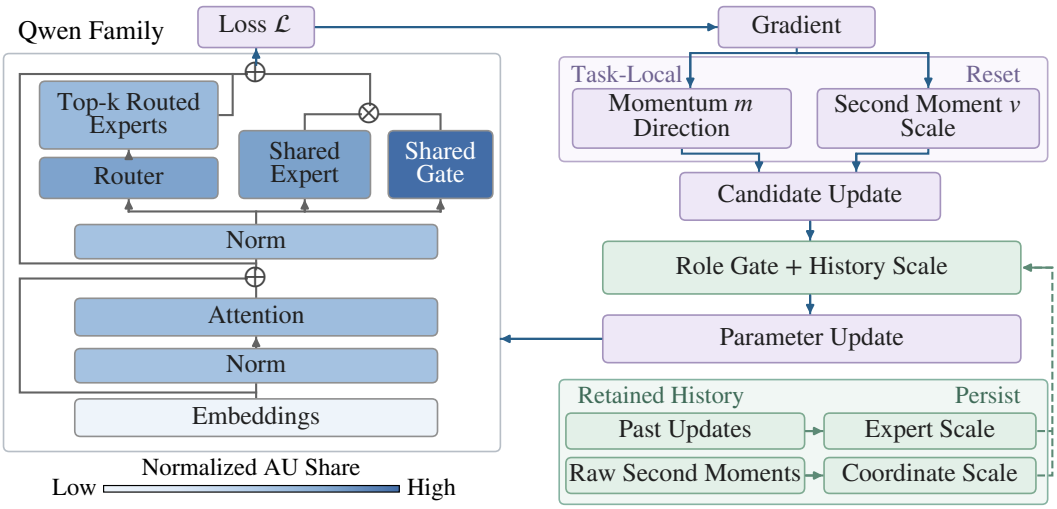


Figure 1: MoRE separates current-task adaptation from retained history. Left: Qwen-family modules colored by their measured accumulated-update shares. Right: task-local moments produce candidate updates; role gates and history scales control the resulting parameter changes.

measurements motivate letting each new task form its own update direction while retaining information about past gradient scales. We then look more closely at expert and module updates, asking where this history accumulates and how it should shape adaptation. At the expert level, task–expert compositions reveal a small task-concentrated tail in OLMoE but more mixed Qwen profiles, suggesting different uses of expert scaling. At the module level, update profiles further distinguish routed from always-active paths, motivating role-level allocation.

Consequently, we propose **MoRE (moment reset and reuse)**, the structured optimizer-update scheme in Figure 1. MoRE resets Adam's fast moments at task boundaries while retaining history to condition subsequent updates. We design role gates to control architectural modules, expert scaling to reuse accumulated update statistics, and coordinate modulation to use slow second-moment history Ω. These controls share one update rule, spanning roles to coordinates without retaining an optimizer for every past task.

We evaluate MoRE on a six-task sequence from the TRACE continual learning benchmark (Wang et al., 2023). On OLMoE-1B-7B (Muennighoff et al., 2024), MoRE averages 49.53 over two repeated runs, versus 45.63 for sequential fine-tuning and 48.39 for EWC; on Qwen1.5-MoE-A2.7B (Team, 2024), its role configuration reaches 54.90, versus 51.98 and 54.60, respectively. Alongside these gains, MoRE avoids a Fisher estimation pass and reduces storage for retained coordinate history by $3\times$ relative to EWC's Fisher-plus-anchor state at equal coverage.

Our contributions are threefold:

- We study boundary-state interventions and update profiles in MoE continual fine-tuning, showing that inherited optimizer state and heterogeneous roles shape both acquisition and retention.
- We propose **MoRE**, separating adaptation from reusable history by making fast moments task-local and reusing cross-task second-moment history to allocate updates from roles to coordinates.
- Evaluating across MoE architectures and LLM and RecSys regimes, we demonstrate our method matches state-of-the-art CL baselines' accuracy at substantially lower training cost.

2 Related Work

Parameter protection and selective updates. Continual learning methods differ in how they preserve earlier capabilities while adapting model parameters. Kirkpatrick et al. (2017) introduce EWC to penalize changes to important weights, with its behavior depending on the Fisher estimator and regularization scale (van de Ven, 2025). Replay methods instead revisit previous examples (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019), while task-dependent masking and parameter isolation

restrict which weights can change (Serrà et al., 2018; Konishi et al., 2023). Momentum-filtered optimization (MoFO) makes this selection using momentum information (Chen et al., 2024). MoRE separates task-local adaptation from history-based update allocation: boundary reset makes fast state task-local, while history-conditioned scaling and role gates act on the resulting Adam update rather than adding an anchor-restoring gradient.

Optimizer-state separation and reuse. Adam uses first and second moments to determine adaptive updates (Kingma & Ba, 2014), and decoupled weight decay separates parameter-side regularization from those recurrences (Loshchilov & Hutter, 2017). Adaptive statistics have also been maintained separately per objective in multi-task learning (Yang et al., 2023), shared and separated in unlearning (Zhong et al., 2025; 2026), and grouped per loss to balance gradient scales in physics-informed networks (Yao et al., 2023). Closest to our setting, ContinualAdam resets task-local moments while reusing a task-size-weighted average of past second moments (Kang & Lee, 2024); we identified this independent and convergent treatment after completing our implementation. Its evaluation targets vision benchmarks with replay over long-tailed streams; our m-only reset arm (Appendix C) evaluates the same reset–reuse design point in few-task LLM fine-tuning, with a continuously evolving second moment in place of a task-size-weighted average. Li et al. (2025) instead recycle Adam's squared-gradient accumulator as a Fisher-diagonal approximation. MoRE combines task-local fast state with history-conditioned update scales and role-to-coordinate allocation.

MoE adaptation and evaluation. MoE continual-learning methods study expert allocation, expert growth, and router–expert consistency (Chen et al., 2023; Li et al., 2024; Xie et al., 2026; Hou et al., 2026; Thérien et al., 2025). Yu et al. (2026) guide selective MoE upcycling with bilevel architecture search and CKA-regularized replay. In contrast to architectural expansion, MoRE studies update allocation within an unchanged architecture and inference procedure. Its architectural roles distinguish routed experts from routers and always-active modules, including shared experts when present. We evaluate OLMoE (Muennighoff et al., 2024) and Qwen on the multi-domain TRACE setting (Wang et al., 2023), using both acquisition and final retention to identify which part of continual adaptation each configuration changes.

3 Understanding Update Interference

We examine how inherited optimizer state and heterogeneous parameter roles affect adaptation, linking acquisition and retention to the reset, reuse, and update-allocation choices in MoRE.

Problem Setup. We study sequential fine-tuning of a fixed-capacity MoE model with parameters θ on a known-boundary task stream $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_T)$. Each task provides a training set $\mathcal{D}_k$ of input–target pairs (x, y) and is visited once, for potentially several epochs. MoRE carries the model parameters forward without replay or task-specific inference modules. Its training objective on the current task is $\mathcal{L}_k(\theta) = -\frac{1}{N_k} \sum_{(x,y)\in\mathcal{D}_k} \sum_{j=1}^{|y|} \log p_\theta(y_j \mid x, y_{<j})$, where $N_k = \sum_{(x,y)\in\mathcal{D}_k} |y|$ is the number of target tokens in task k. We use the six-task TRACE sequence (Wang et al., 2023), following the task selection of Yu et al. (2026): MeetingBank (MB, meeting summarization), Py150 (Python code completion), NumGLUE-cm (cm, commonsense arithmetic), NumGLUE-ds (ds, domain-specific arithmetic), 20Minuten (20Min, text simplification), and C-STANCE (C-ST, stance classification).

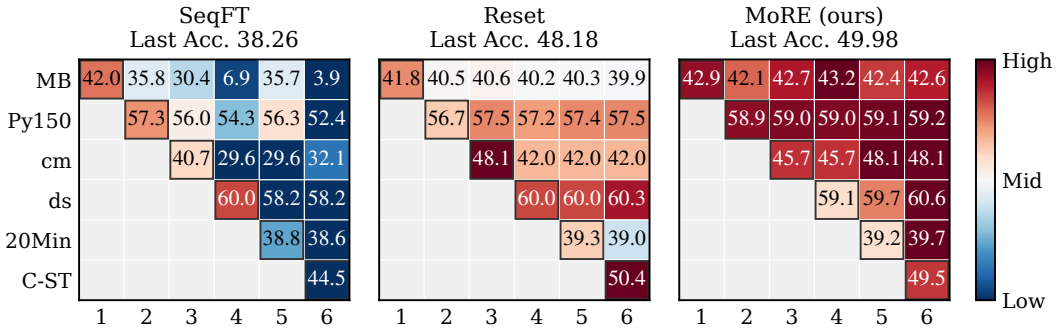


Figure 2: OLMoE task scores under SeqFT (a bad case where the model loses instruction following), reset, and MoRE. Reset preserves MB performance while MoRE gives the highest final mean.

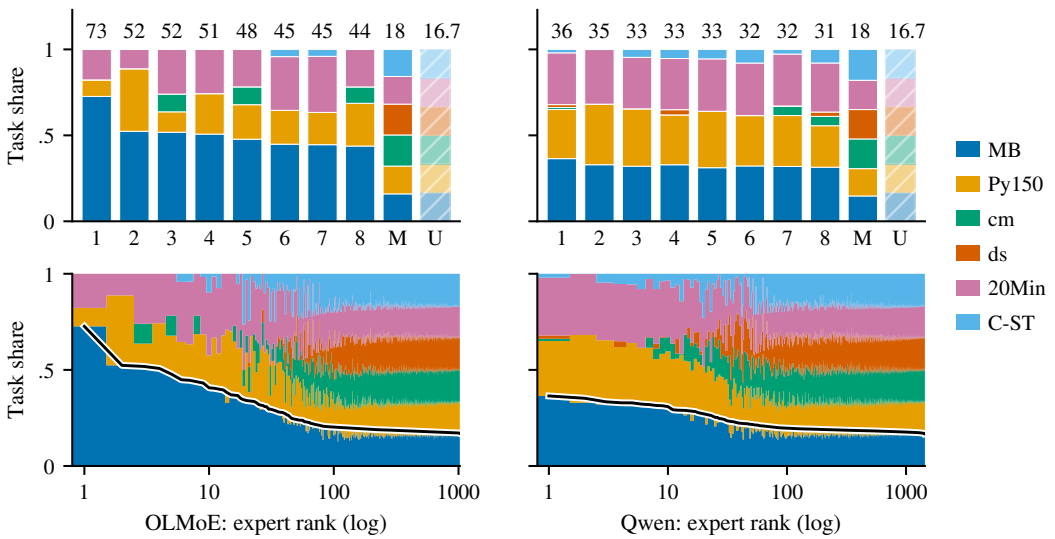


Figure 3: Expert-update compositions measured with MoRE configurations on OLMoE (left) and Qwen (right). Top bars show the eight highest-peak experts, the median-peak expert (M), and a uniform share of $1/6$ (U). Below, the full expert population is ordered by peak share on a logarithmic rank axis. OLMoE has a more task-concentrated expert tail than Qwen.

3.1 Boundary Reset Changes Acquisition and Retention

Successive tasks can require different update directions and scales. A boundary reset provides a direct intervention on inherited optimizer state: it clears the moments while preserving model parameters and the learning-rate schedule. We use this intervention on OLMoE's TRACE sequence to measure how inherited state affects subsequent learning and retention. Figure 2 shows a loss of instruction following on MeetingBank: after SeqFT, the model often answers summarization prompts as fill-in-the-blank questions (e.g., 'C'). Its ROUGE-L falls from 41.96 at acquisition to 3.89 after the stream. Boundary reset preserves summarization performance at 39.93, and adding history-informed expert scaling and role control gives the MoRE run a final average of 49.98.

These OLMoE results motivate boundary reset in the evaluated TRACE recipe, and repeated paired runs confirm that the gain replicates across seeds rather than reflecting a single trajectory (Appendix C). The recommendation stream poses the complementary regime, in which carried moments remain useful (Appendix D). We therefore study which history to reuse rather than prescribe reset for every stream, with selective momentum-based update masking as a direction for further control.

3.2 Fast Adaptation and Expert-Level History Reuse

Resetting adaptation raises a complementary question: which statistics remain useful across tasks? Adam's first moment m tracks the average gradient direction, whereas its second moment v tracks squared gradient magnitudes and sets the adaptive scale (Kingma & Ba, 2014). Our moment-history reuse focuses on v, which retains gradient-scale information without prescribing an update direction. For selective adaptation based on m, MoFO updates the coordinates with the largest momentum magnitudes within each parameter block (Chen et al., 2024). Its mask acts on the parameter update while both moment estimates continue to evolve; testing this selection alongside second-moment reuse is future work. We call the task-local recurrences *fast state* and a separately retained cross-task summary *slow history*. This separation connects task-specific adaptation in AdaTask (Yang et al., 2023) to cross-task second-moment reuse in ContinualAdam (Kang & Lee, 2024). Second moments also connect to importance estimation: both the Fisher diagonal and Adam's v accumulate squared-gradient information, although their gradient sources and averaging differ (Kirkpatrick et al., 2017; Li et al., 2025). Synaptic Intelligence instead uses a gradient–displacement path integral to estimate each parameter's contribution to loss reduction, rather than its update magnitude (Zenke et al., 2017).

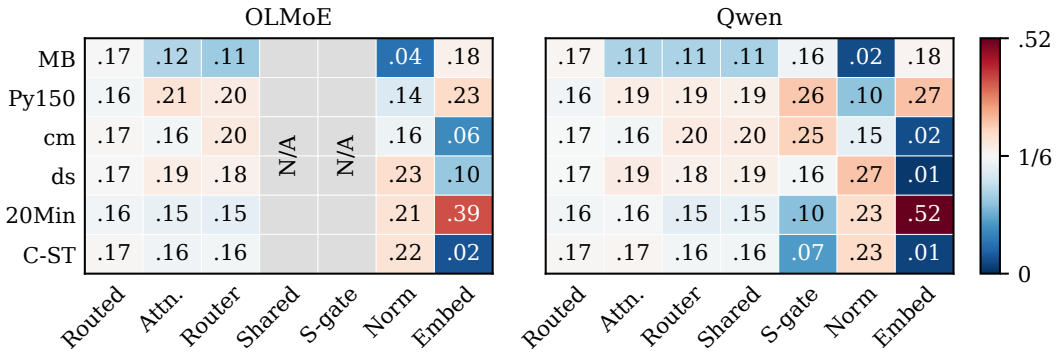


Figure 4: Role-level AU profiles measured with MoRE configurations on OLMoE (left) and Qwen (right), normalized within tasks then across tasks per role. Routed denotes sparse routed experts; Shared/S-gate denote shared experts/their gates, and Norm/Embed normalization/embeddings. Gray marks absent roles. Distinct role profiles reveal update paths that a single model-wide multiplier cannot control separately.

To connect these optimizer statistics to the changes made to each expert, we adopt AdaTask's update-based view of task dominance (Yang et al., 2023). Whereas AdaTask uses an exponentially decaying average of squared updates, our accumulated-update (AU) diagnostic sums squared parameter displacements over measurement windows within each task. Let $\theta_{k,e}^{(b)}$ denote expert e's parameters after measurement window b of task k, with $b = 0$ the task-start snapshot and B_k the number of windows. We define its AU energy as $A_{k,e} = \sum_{b=1}^{B_k} \|\theta_{k,e}^{(b)} - \theta_{k,e}^{(b-1)}\|_2^2$. Whereas v tracks squared gradients before adaptive normalization, AU records squared parameter displacements after the optimizer and update controls act. The two statistics, v and AU, therefore offer complementary views of gradient scale and realized adaptation. Figure 3 applies this diagnostic to MoRE configurations on OLMoE and Qwen, normalizing AU within each task and then across tasks for each expert to show their update compositions.

The compositions suggest different uses of expert-level history. OLMoE has 23 of 1,024 experts with a largest task share of at least $1/3$, versus 3 of 1,440 in Qwen. The same diagnostic applied to an OLMoE EWC run finds 33 of 1,024 experts meeting this threshold, so task concentration also occurs under parameter regularization. This OLMoE tail motivates attenuating later changes to experts shaped disproportionately by earlier tasks; MoRE uses raw-AU history for its expert scales. Qwen's more mixed profiles motivate moving toward role-level control. The plotted task-normalized shares diagnose concentration, while the implemented scales use the raw-AU score defined in Appendix A.

Thus expert AU summaries condition expert scales, while sampled second moments form the coordinate extension's slow history Ω; both survive resets outside the fast recurrences. The coordinate channel retains one BF16 field rather than EWC's FP32 Fisher and BF16 anchor. AU measures displacement at a chosen window and learning-rate schedule, distinct from routing counts, which measure activation.

3.3 Module Profiles Motivate Role-Wise Allocation

Expert-level scaling does not cover the always-active paths that successive tasks also update. OLMoE-1B-7B has 64 routed experts per layer, top-8 routing, and 16 layers (6.92B total parameters). Qwen1.5-MoE-A2.7B has 60 routed experts per layer, top-4 routing, and 24 layers, with an additional always-active shared expert in each layer (14.3B total, 2.7B active). We therefore aggregate AU by routed experts, routers, shared experts and their gates where present, attention, normalization, and embeddings. Figure 4 shows the resulting task profiles.

Different profiles imply that one model-wide multiplier cannot target all update paths independently. EWC's coordinate-wise Fisher-weighted anchor penalty does not explicitly distinguish these architectural roles. MoRE instead gates Qwen's shared experts, attention, and residual dense parameters separately; for OLMoE, it combines expert scaling with a dense-role gate. The profiles motivate this granularity. Section 4.2 makes it operational through a training-dynamics trigger.

Algorithm 1 MoRE with task-local moments and retained history.

Require: Parameters θ, schedule η, gate policy π, enabled history channels
$t, g, A, \Omega \leftarrow 0$; $c \leftarrow 1$
for task $k = 1, \ldots, T$ **do**
  $m, v, V, J \leftarrow 0$; reset bias correction and gate-policy state
  for microbatch $\mathcal{B}_u$ **do**
    Accumulate $\nabla_\theta \mathcal{L}_k(\theta; \mathcal{B}_u)$ into g
    if an optimizer step is due **then**
      $t \leftarrow t + 1$; $m \leftarrow \beta_1 m + (1 - \beta_1) g$; $v \leftarrow \beta_2 v + (1 - \beta_2) g^2$
      $(\hat{m}, \hat{v}) \leftarrow \mathrm{BiasCorrect}(m, v)$; $h_i \leftarrow \pi_{\rho(i)}$ from prior policy state
      $q_i \leftarrow 1 + \kappa \max(\Omega_i - \tau, 0)$ on enabled (k, i), otherwise 1
      $a_i \leftarrow q_i / c_i$; $\xi_i \leftarrow \hat{m}_i / (a_i \sqrt{\hat{v}_i} + \epsilon)$
      Observe pre-gate (ξ, θ) for the next policy state
      $\theta_i \leftarrow \theta_i - \eta_t h_i \xi_i$; $g \leftarrow 0$; advance the schedule
    end if
    Collect enabled AU windows; at valid raw-v samples, $V \leftarrow V + v$, $J \leftarrow J + 1$
  end for
  Flush enabled AU windows; update $c_e \leftarrow (1 + 0.5 s_e(A_{1:k}))^{-1}$ if enabled
  for each enabled group G with $J > 0$ and positive mean signal **do**
    $\bar{v}_i \leftarrow V_i / J$; $\Omega_i \leftarrow \gamma \Omega_i + \bar{v}_i / \mathrm{mean}_{\ell \in G} \bar{v}_\ell$
  end for
end for

4 METHODOLOGY

Guided by these dynamics, MoRE separates task-local adaptation from reusable history, then uses that history to control updates at role, expert, and coordinate levels.

4.1 TASK-LOCAL FAST ADAPTATION AND SHARED HISTORY

MoRE resets Adam's moments (m, v) at each task boundary, while retaining the model parameters, global learning-rate schedule, and cross-task history. For the current task gradient $g_{t,i}$ at optimizer step t and coordinate i, it applies Adam's updates with decay factors β_1 and β_2:

$$m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}, \qquad v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2. \tag{1}$$

Their bias-corrected values are $\hat{m}_{t,i}$ and $\hat{v}_{t,i}$. Fast adaptation therefore starts afresh on each task, while retained history influences the update through the separate controls below.

4.2 STRUCTURED UPDATE

MoRE changes the size of Adam's update through two controls. A gate $h_{t,i} \in [0, 1]$, shared by coordinates in the same architectural role, controls how much that role adapts. A history scale $a_{t,i} > 0$ adjusts adaptation within the role, using expert-level or coordinate-level statistics. With learning rate η_t and numerical stabilizer $\epsilon > 0$, the update is

$$\Delta\theta_{t,i} = -\eta_t h_{t,i} \frac{\hat{m}_{t,i}}{a_{t,i} \sqrt{\hat{v}_{t,i}} + \epsilon}. \tag{2}$$

Increasing a attenuates the update; setting $h = 0$ stops it while the moments continue adapting. Setting both controls to one recovers Adam. Expert history supplies a shared scale within each expert, while coordinate modulation refines it using individual second moments. Appendix A gives the full parameterization.

Shared history corrects the current task's adaptive scale through an implicit shared-plus-task-specific decomposition (Zhong et al., 2026). The effective second moment combines the task-local estimate with a history-conditioned correction,

$$\hat{v}_{t,i}^{\mathrm{eff}} = \hat{v}_{t,i} + \delta_{t,i}^{\mathrm{hist}}, \qquad \delta_{t,i}^{\mathrm{hist}} = (a_{t,i}^2 - 1) \hat{v}_{t,i}. \tag{3}$$

History is shared across tasks through a, while the correction adapts to the current task through $\hat{v}$. Since $\hat{v}^{\text{eff}} = a^2\hat{v}$, MoRE evaluates the corrected denominator as $a\sqrt{\hat{v}} + \epsilon$, without storing an additional residual tensor. It uses retained statistics to scale updates rather than pulling weights toward saved parameters, so no EWC-style parameter anchor is needed.

Gates from training dynamics. We design a causal policy that reduces a role's update when its recent update magnitude exceeds a warm-up reference. It measures the candidate Adam update before gating, relative to the role's weight magnitude. The next step's gate decreases as this ratio rises and returns toward one as it falls. Because the signal is measured before attenuation, the gate cannot suppress its own input. Warm-up and earlier training steps determine the reference and trigger, without test scores or task-specific lookup rules. Appendix A.2 gives the complete policy.

4.3 History-Conditioned Coordinate Modulation

Coordinate modulation uses accumulated second-moment history Ω to dampen coordinates with high gradient activity. Let $\bar{v}_{k,i}$ denote task k's mean raw second-moment and $G(i)$ its role-and-layer group. We normalize each contribution within its group, then use this history on the next task:

$$\Omega_{k,i} = \gamma\Omega_{k-1,i} + \frac{\bar{v}_{k,i}}{\text{mean}_{\ell \in G(i)}\, \bar{v}_{k,\ell}}, \qquad q_{t,i} = 1 + \kappa \max(\Omega_{k-1,i} - \tau, 0). \tag{4}$$

Here γ retains past contributions, τ selects high-history coordinates, and κ controls attenuation. The factor q increases the denominator scale a in Eq. equation 2; it equals one when modulation is disabled. This controls the parameter update without altering Adam's moment-update rule.

For streaming recommendation, history instead accumulates at sampling events while fast moments persist across time periods. Section 5.3 evaluates this continuous variant. Algorithm 1 summarizes the boundary-based procedure. Here V and J are the raw-v sample sum and count, $\rho(i)$ is the coordinate's role, and π supplies its gate from preceding training observations.

5 Experiments

5.1 Experimental Setup

We evaluate whether separating optimizer history and update roles improves task performance, then examine the acquisition, retention, and storage trade-offs behind the results.

Metrics. We measure ROUGE-L for MB, code similarity for Py150, accuracy for cm, ds, and C-ST, and SARI for 20Min, all on a 0–100 scale. Let $M_{r,i}$ be task i's score after training task r, with acquisition $d_i = M_{i,i}$ and final score $f_i = M_{T,i}$. The heatmaps' Last Acc. denotes the final mean $\bar{f}$ across these task-specific metrics. We summarize acquisition, final performance, and backward transfer (BWT) (Lopez-Paz & Ranzato, 2017) as

$$\bar{d} = \frac{1}{T}\sum_i d_i, \qquad \bar{f} = \frac{1}{T}\sum_i f_i, \qquad \text{BWT} = \frac{1}{T-1}\sum_{i<T}(f_i - d_i). \tag{5}$$

Negative BWT indicates deterioration and positive BWT that learning later tasks improves earlier-task performance on average; differences are absolute points.

Models and comparisons. We evaluate the six-task TRACE sequence with OLMoE-1B-7B-0924 (Base), OLMoE-1B-7B-0924-Instruct, and Qwen1.5-MoE-A2.7B. Table 1 groups results by the pre-trained checkpoint. The full-parameter MoRE and EWC runs use three epochs per task, learning rate 1.5×10^{-5}, effective batch size 32, and a global cosine schedule. The Instruct block compares LoRA (Hu et al., 2022) and CKA-NAS selective upcycling (Yu et al., 2026); Appendix A.5 specifies their adapter, replay, and training-budget settings. We calibrate online EWC (Schwarz et al., 2018) at this model scale, using $\lambda = 10^6$ and an empirical Fisher from 400 examples per rank. Independent fine-tuning (IFT) provides a separate-model reference, while sequential fine-tuning (SeqFT) provides the continual baseline. Experience replay(*) revisits the Lima instruction corpus (Zhou et al., 2023) at ratio 0.01 instead of past-task examples. MoFO filters each block's update to its largest-momentum coordinates (Chen et al., 2024); its row comes from a paired comparison at seed 5678 in which both arms branch from a shared MeetingBank-trained prefix and continue over the remaining five tasks, so it is not strictly matched with the full-stream Base rows. MoRE$^-$ is an expert-anchor

Table 1: Final TRACE scores. The OLMoE MoRE and CKA-NAS rows average repeated runs. MoRE gives the highest final mean among the CL methods. BWT (points) averages $f_i - d_i$ over the first five tasks; negative values indicate forgetting. IFT (independent model per task) has no BWT.

Model	Method	MB	Py150	cm	ds	20Min	C-ST	BWT	Avg.
OLMoE-Base	IFT	42.01	60.08	44.44	67.08	39.48	47.20	–	50.05
	SeqFT	39.82	47.19	37.04	60.00	39.21	50.55	-1.51	45.63
	Reset	39.93	57.47	41.98	60.31	39.02	50.40	-1.44	48.18
	Replay(*)	38.80	54.44	34.57	62.77	38.35	40.90	+1.13	44.97
	EWC	41.59	57.74	43.21	57.23	39.09	51.45	-0.75	48.39
	MoFO	40.65	57.71	49.38	58.77	39.22	50.40	+0.44	49.36
	MoRE$^-$ (ours)	42.37	59.06	39.51	65.54	38.69	50.90	+0.00	49.34
	MoRE (ours)	42.17	58.76	46.91	60.62	39.48	49.25	+0.22	**49.53**
OLMoE-Instruct	SeqFT	40.20	57.04	39.51	62.15	38.81	50.20	-2.05	47.99
	LoRA	0.88	53.90	14.81	34.15	37.36	42.70	-16.10	30.63
	EWC	43.45	58.44	43.21	56.31	38.48	49.30	-0.37	48.20
	CKA-NAS	40.18	58.23	43.62	57.95	38.15	49.70	-0.20	47.97
	MoRE (ours)	41.40	58.02	45.68	59.85	38.55	50.25	-0.35	**48.96**
Qwen	SeqFT	40.17	59.64	46.91	67.38	40.89	56.85	+0.23	51.98
	EWC	43.07	58.74	61.73	68.31	39.48	56.25	-0.70	54.60
	MoRE (ours)	42.95	60.59	59.26	69.23	40.81	56.55	-1.51	**54.90**

predecessor that combines boundary reset with AU-weighted penalties on routed experts; it provides an expert-level comparison with MoRE's history-conditioned update scaling and role gates. Table 1 averages two MoRE runs per OLMoE checkpoint and three CKA-NAS runs. The remaining rows report completed individual runs; this variation is small next to the gap between MoRE and SeqFT on OLMoE-Base, so we do not re-measure these baselines.

5.2 Main Results Across MoE Architectures

For OLMoE-Base, reset addresses the largest loss: SeqFT bad case (Fig 2)'s meeting-summary performance collapses as later tasks are learned, whereas resetting moments preserves much of that capability. MoRE then raises the final mean beyond reset alone and closes most of the gap to independent fine-tuning. Replay attains the strongest backward transfer of the block, $+1.13$ points, but a lower final mean than the reset, EWC, and MoRE rows: revisiting general instruction data limits forgetting at the cost of acquisition. MoFO likewise improves over EWC through retention (BWT $+0.44$), at a final mean below MoRE's, with higher memory and time cost. **For OLMoE-Instruct**, MoRE improves on SeqFT and EWC. On this checkpoint, allocating updates within the existing expert pool yields a higher final mean than the evaluated expansion approach with replay and extra task epochs. **For Qwen**, our role-wise control improves on SeqFT and reaches a final mean slightly above EWC's. Its advantages in code completion, domain-specific arithmetic, simplification, and stance classification outweigh lower scores on summarization and commonsense arithmetic. Figure 5 shows that the commonsense-arithmetic difference arises after equal acquisition: EWC preserves more of that task's score. BWT separates the two methods. SeqFT shows no net forgetting on this stream (BWT $+0.23$), and MoRE's -1.51 lies below EWC's -0.70; on streams without net forgetting, the case for boundary control rests on acquisition rather than on preventing forgetting. On Qwen the final-score gain over EWC is modest, but EWC buys its retention with a Fisher-plus-anchor state, an estimation pass, and substantially longer training; MoRE's role-only configuration needs none of these, and the efficiency comparison below quantifies the difference.

Efficiency. The measured OLMoE-Base MoRE run completes training in 4 hours 12 minutes, compared with 10 hours 50 minutes for EWC on the same four-device setup, a $2.6\times$ end-to-end runtime ratio. On Qwen, MoRE completes the stream in 15 hours 31 minutes against 20 hours 49 minutes for EWC, a $1.3\times$ ratio, and SeqFT takes 14 hours 38 minutes. The measured peak GPU memory per rank on Qwen is 65.2 GB for MoRE versus 74.7 GB for EWC. In the paired comparison, MoFO trains $1.5\times$ longer than the boundary-reset MoRE$^-$ arm with 40% higher peak optimizer memory per rank, because it materializes the full update and per-block ranking scores at every step. Separately, analytic storage accounting gives 2 bytes per covered parameter for MoRE's retained coordinate history, versus 6 for EWC's Fisher-plus-anchor state. The $3\times$ reduction applies to re-

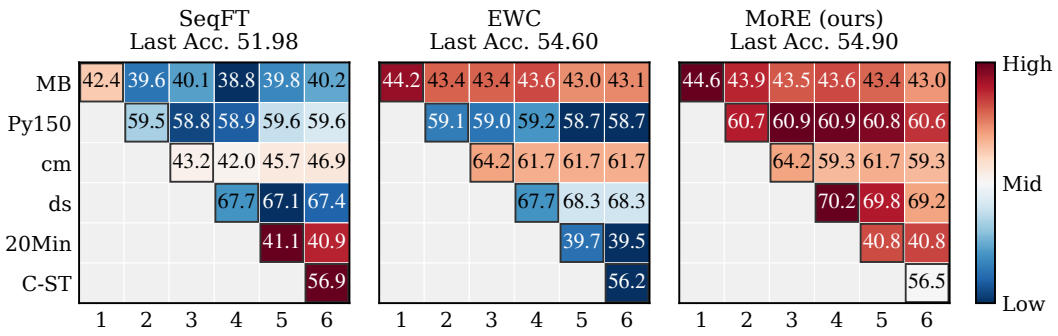


Figure 5: Qwen task trajectories for SeqFT, EWC, and MoRE. Each task uses one shared color scale across methods; outlined cells mark acquisition. MoRE improves the final mean over SeqFT and reaches a similar level to EWC.

Table 2: Final-period AUC on the KuaiRand stream, averaged over the five feedback towers. K counts retained history fields, and the sampling interval counts optimizer steps between history updates. The label-conditioned oracle assigns one field per feedback label and is an upper bound.

Method	K	Sampling interval	Final AUC
SeqFT	–	–	0.8248
Online EWC ($\lambda = 10$)	–	–	0.8309
Replay (10% stored)	–	–	0.8320
MoRE, single field	1	1	0.8355
MoRE, single field	1	50	0.8369
MoRE, routing-conditioned fields	2	50	**0.8392**
Label-conditioned oracle fields	5	50	0.8395

tained history; the FP32 collection accumulator adds 4 bytes per covered parameter during training. Appendix B.2 details the remaining collection and expert-control costs.

5.3 Coordinate Control Under Streaming Drift

TRACE marks its task boundaries explicitly, whereas deployed models often face gradual drift. We therefore test coordinate history on KuaiRand-1K (Gao et al., 2022) with a recommendation MoE: two blocks of eight routed experts plus one always-active shared expert, five jointly trained feedback towers, and four chronological periods of interaction logs. Adam moments persist across periods, and retained history attenuates the update denominator as in Eq. equation 4.

Retained history improves final AUC over sequential fine-tuning, online EWC, and a replay reference that stores 10% of past interactions. Conditioning two fields on routing affinity adds a gain, reaching 0.8392 against the oracle's 0.8395, so a few history fields recover most of what explicit label supervision provides. Appendix D gives the model configuration and a paired state-policy study over 30 seeds. Across seeds, the paired study measures the same channel: history reuse improves backward transfer with and without a boundary reset, while carrying moments outperforms resetting them when the objectives stay fixed and the data distribution drifts.

6 Conclusion

In this work, we asked whether continual MoE fine-tuning should reset or reuse optimizer memory. Measured through boundary-state interventions and update profiles, the answer is both, at different timescales: fast moments are task-local, while second-moment history accumulates where tasks concentrate their updates, in a task-shaped expert tail and in role-specific profiles. We present MoRE, a scheme that acts on this structure, retaining that history to allocate updates from architectural roles to individual coordinates. On the six-task TRACE stream, its evaluated configurations improve final averages over sequential fine-tuning and replay and match parameter regularization, without a Fisher estimation pass and with one third of EWC's retained coordinate state; under recommendation drift, where objectives stay fixed, the same history improves retention while carrying moments outperforms resetting them. Optimizer history is thus a first-class design choice in continual learning: which history to retain, and where it should influence subsequent updates.

AI USE STATEMENT

AI tools assisted with code implementation and review, experiment planning and feedback, conceptual framing, interpretation of diagnostics, and manuscript drafting. The authors remain responsible for verifying technical claims, experimental provenance, citations, and the accuracy and scope of this disclosure.

REFERENCES

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, pp. 556–572. Springer International Publishing, 2018. ISBN 9783030012526. doi: 10.1007/978-3-030-01252-6_33. URL http://dx.doi.org/10.1007/978-3-030-01252-6_33.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. Lifelong language pretraining with distribution-specialized experts. In *ICML 2023*, 2023. URL https://arxiv.org/abs/2305.12281. ICML 2023.

Yupeng Chen, Senmiao Wang, Yushun Zhang, Zhihang Lin, Haozhe Zhang, Weijian Sun, Tian Ding, and Ruoyu Sun. Mofo: Momentum-filtered optimizer for mitigating forgetting in llm fine-tuning. 2024. URL https://arxiv.org/abs/2407.20999.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. KuaiRand: An unbiased sequential recommendation dataset with randomly exposed videos. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022.

Zhiyan Hou, Haiyun Guo, Haokai Ma, Yandu Sun, Yonghui Yang, and Jinqiao Wang. Pass-moe: Mitigating misaligned co-drift among router and experts via pathway activation subspaces for continual learning. *arXiv preprint arXiv:2601.13020*, 2026. ACL 2026.

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

Keller Jordan. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024.

Liwei Kang and Wee Sun Lee. Continual learning of numerous tasks from long-tail distributions. 2024. URL https://arxiv.org/abs/2404.02754.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014. URL https://arxiv.org/abs/1412.6980.

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. In *Proceedings of the National Academy of Sciences*, volume 114, pp. 3521–3526, 2017.

Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. Parameter-level soft-masking for continual learning. 2023. URL https://arxiv.org/abs/2306.14775. ICML2023.

Hongbo Li, Sen Lin, Lingjie Duan, Yingbin Liang, and Ness B. Shroff. Theory on mixture-of-experts in continual learning. *arXiv preprint arXiv:2406.16437*, 2024. ICLR 2025 Spotlight.

Yu Xin Li, Felix Dangel, Derek Tam, and Colin Raffel. Fishers for free? Approximating the Fisher information matrix by recycling the squared gradient accumulator. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34252–34270. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25j.html.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR 2019*, 2017. URL https://arxiv.org/abs/1711.05101. ICLR 2019.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024. ICLR 2025.

Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress compress: A scalable framework for continual learning, 2018. URL https://arxiv.org/abs/1805.06370.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML 2018*, 2018. URL https://arxiv.org/abs/1801.01423. 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. *arXiv preprint arXiv:1701.06538*, 2017.

Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

Benjamin Thérien, Charles Étienne Joseph, Zain Sarwar, Ashwinee Panda, Anirban Das, Shi-Xiong Zhang, Stephen Rawls, Sambit Sahu, Eugene Belilovsky, and Irina Rish. Continual pre-training of moes: How robust is your router? 2025. URL https://arxiv.org/abs/2503.05029.

Gido M. van de Ven. On the computation of the fisher information in continual learning. In *ICLR 2025*, 2025. URL https://arxiv.org/abs/2502.11756. ICLR 2025.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. Trace: A comprehensive benchmark for continual learning in large language models. 2023. URL https://arxiv.org/abs/2310.06762.

Zhen-Hao Xie, Jun-Tao Tang, Yu-Cheng Shi, Han-Jia Ye, De-Chuan Zhan, and Da-Wei Zhou. Same: Stabilized mixture-of-experts for multimodal continual instruction tuning. *arXiv preprint arXiv:2602.01990*, 2026. ICML 2026.

Enneng Yang, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo. Adatask: A task-aware adaptive learning rate approach to multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10745–10753, 2023. doi: 10.1609/aaai.v37i9.26275.

Jiachen Yao, Chang Su, Zhongkai Hao, Songming Liu, Hang Su, and Jun Zhu. MultiAdam: Parameter-wise scale-invariant optimizer for multiscale training of physics-informed neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 39702–39721. PMLR, 2023. URL https://proceedings.mlr.press/v202/yao23c.html.

Zhiyuan Yu, Enneng Yang, Hao Jiang, Guojie Zhu, Feihong He, Peng Wang, and Li Shen. Efficient bilevel optimization for cka-guided moe upcycling. In *International Conference on Machine Learning*, 2026.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML 2017*, 2017. URL https://arxiv.org/abs/1703.04200. ICML 2017.

Xuyang Zhong, Haochen Luo, and Chen Liu. Dualoptim: Enhancing efficacy and stability in machine unlearning with dual optimizers. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Xuyang Zhong, Qizhang Li, Yiwen Guo, and Chen Liu. Dualoptim+: Bridging shared and decoupled optimizer states for better machine unlearning in large language models. *arXiv preprint arXiv:2605.21539*, 2026. ICML 2026.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems*, 2023.

Limitations and Future Work

The contrast between TRACE task changes and recommendation drift motivates selective update control rather than a universal reset rule. Future work will combine second-moment reuse with MoFO-style momentum-based update masking and test training-dynamics gates across further MoE families, datasets, and longer streams. Muon provides another direction through matrix updates that use orthogonalized momentum without Adam's second-moment accumulator (Jordan, 2024). Extending history control to those updates therefore calls for another signal, such as a squared-gradient proxy or a gradient–displacement path integral, together with tests of the placement and cost of update masks. Appendix C.2 reports the existing optimizer comparison that motivates this study.

A Implementation and Configurations

Table 3 reports the practical gate values of the evaluated configurations. The warm-up policy in Appendix A.2 derives the allocation forward from pre-gate update magnitudes observed during training, without test scores or task-specific lookup rules.

A.1 Role Gates

A floor is the endpoint of a decaying gate, or its value throughout a segment when the segment-constant flag is one. Attention and shared-expert settings override the residual dense setting. Routed experts and routers retain unit role gates unless specified otherwise; the OLMoE expert multiplier is applied in addition to the role gate.

For L_k optimizer steps in a segment and local step ℓ, let $s = \min(1, \ell/L_k)$. The decaying and segment-constant gates are, respectively,

$$h_{k,\rho}(s) = 1 - (1 - \phi_{k,\rho})s, \qquad h_{k,\rho}(s) = \phi_{k,\rho}. \tag{6}$$

The global cosine schedule continues across segments. A zero decaying endpoint therefore differs from a constant zero gate, which suppresses the role's updates throughout the segment.

Table 3: Practical role-gate configurations in task order. OLMoE enables raw-AU expert scaling with $\alpha = 0.5$; Qwen uses $c_i = 1$. The configurations allocate different levels of protection to always-active parameter roles.

Configuration	Setting	MB	Py	cm	ds	20Min	C-ST
MoRE (OLMoE)	Dense floor (including attention)	1	1	0	1	1	1
MoRE (OLMoE)	Segment-constant flag	0	0	0	0	0	0
MoRE (Qwen)	Residual dense floor	1	1	0	1	1	1
MoRE (Qwen)	Shared-expert floor	1	1	0	0	0	1
MoRE (Qwen)	Attention floor	1	1	0	0	0.5	1
MoRE (Qwen)	Segment-constant flag	0	0	1	1	1	0

A.2 Training-Dynamics Gates and Expert Scores

Let I_ρ contain the coordinates of architectural role ρ, and let $\xi_{t,i} = \hat{m}_{t,i}/(a_{t,i}\sqrt{\hat{v}_{t,i}} + \epsilon)$ be the candidate update before applying the gate or learning rate. The policy measures relative update magnitude, smooths it, and uses the previous step's value to set the next gate:

$$z_{t,\rho} = \frac{\sum_{i \in I_\rho} \xi_{t,i}^2}{\sum_{i \in I_\rho} \theta_{t-1,i}^2 + \epsilon_\rho}, \qquad \bar{z}_{t,\rho} = \beta_h \bar{z}_{t-1,\rho} + (1-\beta_h) z_{t,\rho}, \\ h_{t,\rho} = \min\left(1, \sqrt{b_{k,\rho}/\bar{z}_{t-1,\rho}}\right). \tag{7}$$

Here $\epsilon_\rho > 0$ stabilizes the relative scale and $0 \le \beta_h < 1$ controls smoothing; the first observation initializes $\bar{z}$. Gates stay at one during the first W_h optimizer steps of each task. The p_h-quantile of the warm-up observations, with $0 < p_h < 1$, sets $b_{k,\rho}$ for the rest of that task. A zero reference or demand leaves the gate at one. If the task ends before warm-up completes, its gates remain at one throughout.

Rising demand lowers the gate; falling demand releases it. The lagged observation makes the decision causal, and measurement before gating avoids directly suppressing the trigger through attenuation. Warm-up length, quantile, and smoothing are shared across tasks.

Expert history. The OLMoE expert score uses only AU collected through the preceding task boundary: it uses neither future-task observations nor task test scores. Before the first boundary, $c_e = 1$. After r tasks, let $A_{j,e}$ be expert e's raw AU on task j. With one observed task, set $b_e = A_{1,e}$. With multiple tasks and positive total AU, define

$$p_{j,e} = \frac{A_{j,e}}{\sum_{u=1}^{r} A_{u,e}}, \qquad b_e = p_{(1),e} - p_{(2),e}, \qquad s_e = \frac{b_e}{\mathrm{mean}_{e'}\, b_{e'}}, \qquad c_e = (1 + 0.5 s_e)^{-1}, \tag{8}$$

where $p_{(1),e}$ and $p_{(2),e}$ are the largest two shares. An expert with zero total AU has $b_e = 0$; an empty or zero-mean score history leaves $c_e = 1$. This score does not first normalize each task's total AU. It is therefore distinct from the doubly normalized profiles used for display in Appendix C.6. Expert AU uses 64-microstep windows, with 15 windows per task including the final flush; the shorter router diagnostic is not used to set c_e.

A.3 Coordinate History

The boundary-based coordinate variant in Table 9 uses the Qwen gates in Table 3. Raw second moments are sampled every 100 task microsteps from the first task. Normalization groups are defined by role and layer and cover routed experts, routers, and always-active parameters. This variant uses three epochs per task, learning rate 1.5×10^{-5}, and CPU-resident optimizer state.

For J_k valid samples in task k, let u_j be the sampled microstep and $t(u_j)$ the most recent optimizer step. The mean used in Eq. equation 4 is

$$\bar{v}_{k,i} = J_k^{-1} \sum_{j=1}^{J_k} v_{t(u_j),i}. \tag{9}$$

Sampling reads the latest raw second moment after each selected microbatch, including microbatches without a new optimizer step. It skips the initial zero state. A group is folded only when samples exist and its mean is positive; otherwise its history remains unchanged. Every valid normalized contribution has group mean one. The global learning-rate schedule continues across tasks, independently of sampling.

For coordinate i in routed expert e, the complete denominator scale is $a_{t,i} = q_{t,i}/c_e$. Expert history sets a common attenuation within an expert, and q adds coordinate-specific attenuation. Unused history channels have unit controls. Bias correction uses the number of participating optimizer steps since the last reset; this counter resets with the moments, while an m-only reset leaves it and v unchanged.

Boundary semantics. Collection reads the latest optimizer state after a microbatch and skips the initial all-zero state. Boundary reset clears the first and second moments and their participating-step counters; the global learning-rate schedule continues. Sampling cadence and optimizer-step cadence are distinct. A segmented continuation must preserve the specified accumulation remainder, parameter precision, and scheduler state to match uninterrupted execution.

A.4 Training and Evaluation Settings

The full-parameter MoRE and EWC comparisons use three epochs per task, learning rate 1.5×10^{-5}, effective batch size 32, zero weight decay, gradient clipping at 1, and an uninterrupted global cosine schedule. Training uses ZeRO-2 with four distributed ranks. Qwen uses prompt/answer limits of $512/256$ tokens, microbatch size 4 per rank, and gradient accumulation of 2. Online EWC uses $\lambda = 10^6$, an FP32 empirical Fisher estimated from 400 examples per rank, and BF16 anchors.

Table 4: Per-seed statistics of the repeated OLMoE rows in Table 1. MoRE averages two runs per checkpoint and CKA-NAS three; every cell reports the mean ± standard deviation across seeds.

Model	Method	MB	Py150	cm	ds	20Min	C-ST	BWT	Avg.
OLMoE-Base	MoRE (ours)	42.17±0.64	58.76±0.63	46.91±1.75	60.62±0.00	39.48±0.36	49.25±0.42	+0.22±1.00	**49.53±0.63**
OLMoE-Instruct	CKA-NAS	40.18±0.19	58.23±0.25	43.62±3.56	57.95±2.27	38.15±0.30	49.70±0.22	-0.20±1.18	47.97±0.49
	MoRE (ours)	41.40±0.36	58.02±0.02	45.68±3.49	59.85±1.52	38.55±0.12	50.25±0.57	-0.35±0.62	**48.96±0.89**

The OLMoE MoRE rows in Table 1 average two executions per checkpoint; Table 4 reports their per-cell statistics, including the three-run CKA-NAS comparison. Figure 2 displays one complete MoRE trajectory, with final average 49.98, rather than the table's two-run mean of 49.53. Each block of Table 1 is scored under a single evaluation chain, and the Qwen rows share the batched expert evaluator with the reset-policy triplet in Appendix C. Heatmaps print absolute scores and share a color scale within each task across methods; colors are not comparable across tasks. Color endpoints use the pooled observed range, with its midpoint as the default white point. Figure 2 places the MB and ds white points at 40.5 and 58.75; Figure 5 uses a 0–100 cm range with white at 40.

A.5 Instruct Benchmark

The Instruct comparison uses OLMoE-1B-7B-0924-Instruct and evaluates 20Minuten with SARI. LoRA uses rank-8 adapters on attention projections and evaluates the merged adapters. The CKA-NAS expert-expansion comparison (Yu et al., 2026) averages three executions: eight experts are added per task in ten selected layers, with top-8 routing, expert learning-rate multiplier 6.67, and replay weight 0.5. Its task epoch schedule is $(6, 3, 3, 3, 3, 8)$, compared with three epochs per task for full-parameter MoRE. This comparison combines expert expansion, replay, and additional acquisition training rather than isolating a single change to the update rule.

B Update Identities and Storage

We show why a single-visit task stream does not require a bank of task-specific Adam states, and why shared history can control the effective learning rate without storing additional residual moments.

B.1 Reset and Effective-Rate Identities

Single-visit state equivalence. Consider two Adam executions that begin each task with identical weights. One resets its moments and counters at each boundary; the other lazily creates zero-initialized state for each new task. If tasks are visited once and gradients, hyperparameters, learning-rate schedules, update masks, parameter partitions, and accumulation semantics agree, their within-task iterates agree. At the first step their moments, counters, and accumulated gradients coincide; identical Adam updates preserve this equality inductively. Thus resetting one state pair suffices in place of retaining unused past-task states. Revisiting a task with its saved moments is a different operation.

History as effective-rate control. For $a_{t,i} = q_{t,i}/c_{t,i} > 0$ and $\hat{v}_{t,i} \geq 0$, Eq. equation 3 gives $\hat{v}^{\text{eff}}_{t,i} = a^2_{t,i}\hat{v}_{t,i}$. At finite $\epsilon > 0$,

$$\sqrt{\hat{v}^{\text{eff}}_{t,i}} + \epsilon = a_{t,i}\sqrt{\hat{v}_{t,i}} + \epsilon, \qquad \Delta\theta_{t,i} = -\eta_t h_{t,i} \frac{\hat{m}_{t,i}}{\sqrt{\hat{v}^{\text{eff}}_{t,i}} + \epsilon}. \tag{10}$$

Equivalently, the multiplier on the ordinary Adam update is

$$r_{t,i} = h_{t,i} \frac{\sqrt{\hat{v}_{t,i}} + \epsilon}{a_{t,i}\sqrt{\hat{v}_{t,i}} + \epsilon}, \qquad \Delta\theta_{t,i} = -\eta_t r_{t,i} \frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}} + \epsilon}. \tag{11}$$

This multiplier approaches $h_{t,i}/a_{t,i}$ when the ϵ terms are negligible; $h_{t,i} = 0$ stops the update exactly. The history-conditioned correction is evaluated implicitly, without a separate tensor or EMA. These are same-step identities in real arithmetic, not equivalence to arbitrary task-specific moments: a nonnegative gate cannot reverse the sign of an update when a different first moment would do so.

Why gradient scaling is different. A constant $a > 0$ applied to every gradient in a coordinate's history, with matched counters and initial states $m'_0 = am_0$, $v'_0 = a^2 v_0$, yields $\hat{m}'_t = a\hat{m}_t$ and $\hat{v}'_t = a^2\hat{v}_t$. Adam then cancels this scaling when $\epsilon = 0$ and the denominator is positive; finite ϵ instead behaves as ϵ/a. Time-varying masks do not generally preserve these relationships. Nor does EWC: its added gradient $\lambda\mathcal{F}_i(\theta_i - \theta^\star_i)$ is not generally a fixed positive multiple of the task gradient. The cancellation identity therefore does not apply to general EWC, projection, or soft masking.

B.2 Storage and Computation

Table 5 counts additional tensor payload beyond model weights, gradients, and ordinary Adam state. Let N be the common coordinate coverage, W the number of balanced optimizer partitions, N_{shadow} the AU shadow size, E the expert count, and R the number of roles.

Table 5: Analytic additional payload with BF16 coordinate history and anchors, and FP32 collection accumulators and Fisher estimates. AU shadows reside on the CPU independently of optimizer sharding; temporary buffers are excluded. Coordinate history is $3\times$ smaller than EWC's retained state at equal coverage, while its collection payload is $6N/W$ bytes.

Component	Retained	During collection
Boundary reset	None	None
Prescribed role gates	$O(TR)$ scalars	$O(TR)$ scalars
Dynamics gates	$O(R)$ scalars	$O(RW_h)$ scalars
AU control	$2N_{\text{shadow}}$ bytes + summaries	$2N_{\text{shadow}}$ bytes + summaries
Coordinate history	$2N/W$ bytes	$6N/W$ bytes
Online EWC	$6N/W$ bytes	$6N/W$ bytes + temporaries

Retained versus collection state. One BF16 history field retains $2N/W$ bytes per rank. The FP32 within-task accumulator adds $4N/W$, making $6N/W$ during collection, equal to EWC's FP32 Fisher plus BF16 anchor before temporaries. The $3\times$ reduction concerns retained history, not peak memory. Offloading changes placement rather than total payload. A bank of K task-specific FP32 moment pairs would require $8KN/W$ bytes; adding a BF16 anchor per task gives $10KN/W$. This

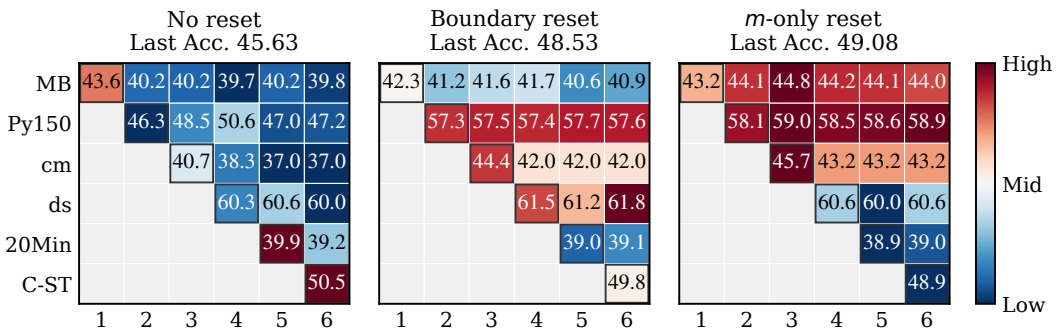


Figure 6: OLMoE state-policy comparison: no reset, boundary reset, and m-only reset. Columns index completed tasks, with a common observed score range per task across panels and outlined acquisition cells. Both reset policies improve the final mean in these runs, with the largest gains on MB and Py150.

latter comparator is a state bank with anchors, not canonical AdaTask. MoRE's coordinate-history size is independent of task count.

Additional work. Reset clears existing state in $O(N)$ work without an extra forward/backward pass. Role gates broadcast scalar multipliers; the dynamics policy also reduces candidate-update norms during training. AU control maintains $O(TE)$ summaries, copies weights, and reduces squared displacements at collection windows. Coordinate control samples existing second moments, reduces group means, and folds history at boundaries. EWC additionally estimates the Fisher and computes its anchor penalty. These payload and operation counts do not substitute for matched peak-memory and throughput measurements.

Implementation checks. Operator checks cover selective state clearing, retention of independent history, counter behavior, no-reset invariance, EWC penalty gradients and sharding, and expert/role scale mapping. Scalar checks separately verify the effective-denominator and effective-rate identities. Distributed trajectory equivalence also requires matching parameter partitions, numerical precision, and boundary restoration semantics.

C Ablations and Diagnostics

C.1 Reset Components

Table 6 and Figure 6 compare no reset, full boundary reset, and first-moment-only reset over six tasks.

Table 6: Six-task state-policy comparison; BWT averages the first five tasks. Initial MB acquisition precedes any boundary intervention. Both reset policies improve final performance, with gains in acquisition as well as retention.

Boundary policy	Final avg.	Mean acquisition	BWT	Initial MB
No reset	45.63	46.89	−1.51	43.61
Boundary reset	48.53	49.05	−0.63	42.31
m-only reset	49.08	49.22	−0.17	43.18

Full reset clears m, v, and the participating-step counters used for bias correction. The m-only policy clears the first moment while retaining the second moment and counters, this arm is the closest evaluated point to ContinualAdam's boundary treatment (Kang & Lee, 2024); neither restarts the global learning-rate schedule. Its result is consistent with useful gradient-scale history surviving a boundary even when inherited momentum is cleared.

Five additional within-protocol repetitions comparing full reset with no reset give a mean final-score difference of $+0.69$ points, with four of the five paired differences positive and the remaining one the smallest in magnitude; the mean acquisition and BWT differences are $+0.65$ and $+0.05$ points. The reset gain thus replicates in direction across seeds, not only in the single displayed trajectory. These repetitions evaluate boundary-state control, not the role-gate policy.

C.2 Muon Expert Updates

We compare ordinary AdamW with a hybrid optimizer that applies Muon to routed-expert matrices while retaining AdamW on other parameters (Jordan, 2024). Both train the six-task TRACE sequence with unit role gates and no AU scaling, anchor penalty, or coordinate modulation. Base and Instruct checkpoints are evaluated separately with the same batched expert evaluator. Table 7 gives the complete optimizer comparison, separate from the main-table SeqFT executions.

Table 7: Expert-matrix optimizer comparison on TRACE; scores use the 0–100 scale. Muon improves mean acquisition on Base but has worse BWT; on Instruct, both acquisition and BWT decline.

Checkpoint	Optimizer	Final mean	Acquisition	BWT
OLMoE-Base	AdamW	48.34	48.83	-0.59
OLMoE-Base	Muon + AdamW	47.70	49.61	-2.29
OLMoE-Instruct	AdamW	49.56	50.22	-0.79
OLMoE-Instruct	Muon + AdamW	46.55	48.99	-2.94

A separate same-hardware timing probe measures 0.897 seconds per step for AdamW and 3.169 for the Muon hybrid, a $3.53\times$ ratio. Matrix orthogonalization is a substantial cost in this implementation. Since Muon does not retain Adam's v on its expert matrices, applying history-based control there requires a different statistic rather than reusing the present coordinate rule unchanged.

C.3 Local Learning-Rate Controls

Paired Qwen restarts test uniform attenuation during one task. Halving the learning rate reduces ds acquisition by 5.54 points in both the initial and fresh-process evaluations, a net loss of 18 correct answers out of 325 (exact paired McNemar $p = 0.0064$). The smaller cm acquisition contrast is unresolved by the paired test. This comparison therefore shows a reproducible ds acquisition cost without a consistent cm retention gain.

Table 8: Initial evaluation scores for paired task-local restarts. A fresh-process repeat preserves both ds scores; half-rate cm retention changes from $51/81$ to $50/81$. Uniform attenuation incurs a reproducible ds acquisition cost, while the cm acquisition difference is unresolved by the paired test.

Learning rate	cm acquisition	ds acquisition	cm after ds
Full rate	60.49	71.38	62.96
Half rate	64.20	65.85	62.96

C.4 Role-Gate Configurations

Table 9 compares three Qwen configurations with increasingly role-specific gates. They differ in gate placement and expert scaling. The role-gated configuration has the highest final mean and ds final score, while the attention-gated configuration has higher cm and C-ST final scores.

C.5 Expert Anchoring

MoRE$^-$ resets optimizer state and trains all parameters with an AU-weighted penalty on routed-expert displacement,

$$\mathcal{L} = \mathcal{L}_t + \frac{1}{2}\sum_{e\in\mathcal{E}} \lambda_0 s_e \|\theta_e - \theta_e^\star\|_2^2, \qquad \lambda_0 = 10^{-2}. \tag{12}$$

Table 9: Qwen configuration-level final scores. Gate placement and expert scaling vary across configurations. Role-wise allocation improves the final mean, but its task-specific trade-offs differ from attention-only control.

Qwen configuration	MB	Py150	cm	ds	20Min	C-ST	Avg.
Dense gate	43.82	58.68	55.56	65.85	41.07	56.20	53.53
Attention gate	42.30	59.49	60.49	65.54	41.19	58.05	54.51
Role gates	42.95	60.59	59.26	69.23	40.81	56.55	54.90

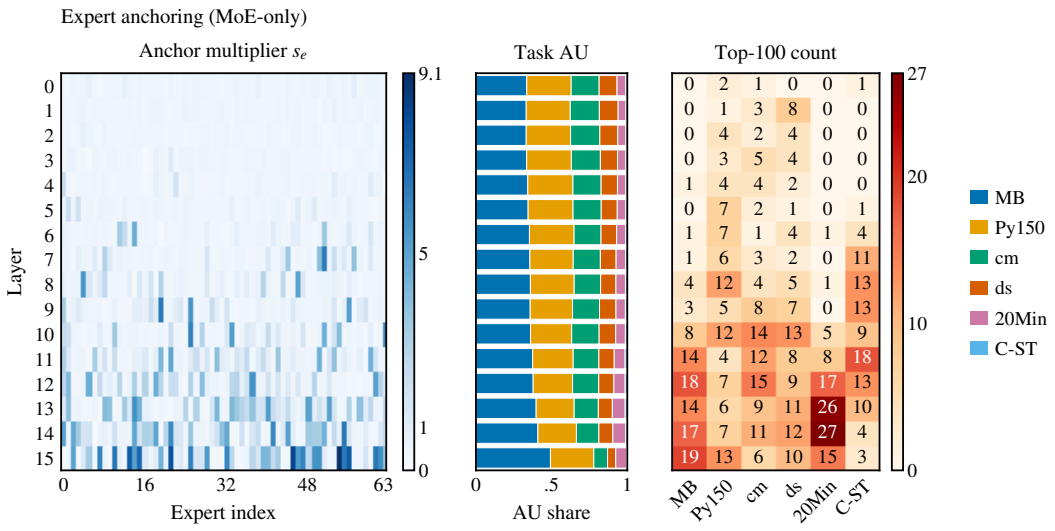


Figure 7: Layer-wise structure of the MoE-only anchoring variant on OLMoE-Base. Left: normalized anchor multipliers s_e reconstructed after all six tasks, with $\lambda_e = 0.01 s_e$. Center: each layer's task composition of raw AU. Right: layer counts among the 100 largest-AU experts for each task. All panels use the same accumulated squared-displacement record. Anchor strength increases toward deeper layers, while the high-AU experts occupy different depths across tasks.

Here $\theta_e^\star$ is the preceding boundary snapshot and s_e is the raw-AU score in Appendix A.2. Unlike EWC's coordinate-wise Fisher weights, this penalty uses one multiplier per expert. Full MoRE replaces the anchor penalty with structured update scaling.

Figure 7 examines a separate MoE-only anchoring configuration, which trains routed experts and routers. Its 16×64 expert grid is reconstructed from all six tasks' raw-AU history using the same largest-share-gap normalization. These final-boundary multipliers are a diagnostic, not gates fixed throughout training. Mean strength rises from 0.43 in the first layer to 2.87 in the last, revealing depth-dependent protection.

C.6 Update and Routing Diagnostics

Displacement and routing. For task displacement $\Delta^{(k)} = \theta^{(k)} - \theta^{(k-1)}$ and total displacement $D = \theta^{(T)} - \theta^{(0)}$, the first-task projection ratio $\langle \Delta^{(1)}, D \rangle / \|\Delta^{(1)}\|^2$ is 1.0047 for experts, 1.0194 for routers, and 0.9990 for attention. MeetingBank token-wise routing overlap changes by 2.95 percentage points after its own round. Near-preserved displacement projections and modest routing changes can therefore coexist with functional performance loss; these statistics do not resolve nonlinear representation changes.

AU normalization. Figures 4 and 3 measure the OLMoE MoRE configuration and the dense-gate Qwen configuration in Table 9. The Qwen diagnostics therefore characterize that gate configuration, rather than the final role-gated setting. The OLMoE EWC comparison uses the same 64-microstep

AU windows, with 15 windows per task. For expert AU $A_{k,e}$, the plotted task shares are

$$U_{k,e} = \frac{A_{k,e}}{\sum_{e'} A_{k,e'}}, \qquad S_{k,e} = \frac{U_{k,e}}{\sum_j U_{j,e}}, \qquad \text{peak}(e) = \max_k S_{k,e}. \tag{13}$$

Role profiles apply the same two-stage normalization after aggregating AU by role, with the first denominator summing all roles. Zero-signal entries are left at zero; absent roles are masked. Under this display normalization, EWC has 33 of 1,024 experts above the $1/3$ peak-share threshold. The raw expert-control score instead normalizes each expert's accumulated history directly, retaining differences in total AU between tasks.

Some task-concentrated OLMoE experts have only 0–5% later-task update shares and total displacement 0.55–0.78 times the median expert, so low subsequent movement contributes to their concentrated profiles.

Scope of update diagnostics. A fixed-weight, cached-gradient comparison isolates an update operator, whereas closed-loop training includes the gradients induced by subsequent updates. Permuting history within a group preserves its values but not their association with update coordinates; equal history distributions therefore need not produce equal update norms. Trajectory diagnostics based on peak-to-final drop also differ from BWT, which references acquisition rather than the best later score (Chaudhry et al., 2018).

D Streaming Recommendation

D.1 Dataset and Model

KuaiRand-1K is divided into four chronological periods of 400,000 standard-log interactions, each with a fixed holdout of 100,000 interactions; random-exposure logs are excluded and vocabulary entries require at least ten occurrences. The five feedback towers for click, like, comment, forward, and negative feedback are trained jointly in every period, so periods form a drift axis rather than a sequence of separate tasks. Table 10 lists the model and training configuration. Table 2 uses one run per configuration and scores the final period's holdout.

Table 10: Configuration of the streaming recommendation model and its training. Each routed expert is a full two-layer network, and routing is shared across feedback objectives.

Setting	Value
Input	User/video ID embeddings and four numeric columns: duration, play ratio, hour, and date
Shared trunk	Concatenated inputs $\to 256$, GELU
MoE blocks	Two blocks; each has eight routed experts and one always-active shared expert
Expert networks	$256 \to 512 \to 256$, GELU between linear layers
Routing	One $256 \to 8$ router per block, shared by all towers; softmax top-2, selected weights renormalized to sum to one
Prediction towers	Five $256 \to 128 \to 1$ networks; GELU hidden activation and sigmoid output
Training loss	Mean binary cross-entropy over all five labels + 0.01 times summed load-balancing losses
Optimizer	AdamW; constant learning rate 10^{-3}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-5}
Training budget	Batch size 4096; one pass (98 optimizer steps) per period
History control	$K = 1$, collection every step, $\kappa = 0.5$, $\gamma = 1$; attenuation begins after shared first period
Threshold	First-period 0.97 quantile of the first 10^6 flattened history entries in parameter order
Online EWC	$\lambda = 0.3$; squared batch-gradient estimate from 500 sampled training examples per period

D.2 State Policies Across Paired Runs

A second study repeats the stream for 30 random initializations. Within each pair, all arms branch from the same first-period weights, Adam state, and accumulated history, and then receive the same minibatch order, so differences between arms are not confounded by data order. Every end-of-period model is scored on all four holdouts: Final averages the four holdouts after the last period, Acquisition averages each holdout after its own period, and BWT averages their difference over the first three periods.

Carryover keeps m, v, and the step counter across boundaries, full reset clears all three, and m-only reset clears the first moment. Adding Ω applies history-conditioned attenuation on top of either state policy. Online EWC uses $\lambda = 0.3$ with the estimator in Table 10. Replay^{+}+ strengthens the replay reference of Table 2: it stores 10% of each period's rows and trains on an additional 4096-example minibatch at every step, doubling the data processed per update. Replay^{++} thus stands outside the equal-exposure comparison as a data-advantaged reference and attains the study's strongest retention (BWT $+0.0270$); at fixed data exposure, history reuse is the component that improves retention.

Table 11: State policies over 30 paired runs: mean $\pm$ standard deviation across seeds, averaged over the five feedback towers.

State policy	Final AUC	Acquisition AUC	BWT
Carryover	0.8282 ± 0.0092	0.8272 ± 0.0060	$+0.0014 \pm 0.0123$
Full reset	0.8189 ± 0.0085	0.8229 ± 0.0064	-0.0054 ± 0.0129
m-only reset	0.8236 ± 0.0094	0.8262 ± 0.0057	-0.0034 ± 0.0132
Carryover+Ω	0.8342 ± 0.0082	0.8272 ± 0.0058	$+0.0094 \pm 0.0098$
Full reset+Ω	0.8273 ± 0.0081	0.8250 ± 0.0059	$+0.0031 \pm 0.0127$
Online EWC	0.8274 ± 0.0089	0.8272 ± 0.0058	$+0.0003 \pm 0.0119$
Replay^{+}+	0.8481 ± 0.0068	0.8279 ± 0.0056	$+0.0270 \pm 0.0077$

History reuse improves retention under both state policies (Table 11): BWT rises by 0.0080 over carryover and by 0.0085 over full reset, with unchanged acquisition. Most of this gain comes from the negative-feedback objective, the only tower that forgets under carryover, whose holdouts contain 24–43 positive examples. Replay^{+}+ reaches the highest final AUC in the study by doubling the data processed at each update; at fixed data exposure, history reuse is the component that improves retention. Unlike the TRACE sequence, full reset here lowers both acquisition and retention relative to carryover, and combining reset with Ω does not recover that loss. Resetting fast state pays off when a boundary changes the objective, whereas carried second moments stay useful when the same objectives drift.

D.3 History Fields

For sampling event j, let $P(i)$ be the parameter tensor containing coordinate i, and let v_j be its raw Adam second moment before the current update. The single-field history is

$$\Omega^{\text{stream}}_{j,i} = \gamma \Omega^{\text{stream}}_{j-1,i} + \frac{v_{j,i}}{\max\big(\text{mean}_{\ell \in P(i)}\, v_{j,\ell}, 10^{-30}\big)}. \tag{14}$$

This implementation normalizes within each parameter tensor; the denominator floor makes a tensor with zero mean signal contribute zero instead of an undefined ratio. History persists across every boundary, whether Adam moments are carried or reset. The threshold τ is the first period's 0.97 quantile over the first 10^6 history entries in parameter order, and attenuation then divides by $\sqrt{\hat{v}_i}\{1+\kappa \max(\Omega_i - \tau, 0)\} + \epsilon$.

With $K > 1$, fields are indexed by routing affinity rather than by experts or towers. Let $w_{j,k} \geq 0$ with $\sum_k w_{j,k} = 1$ be the current batch's affinity to field k. Each field accumulates its affinity-weighted share, and the readout averages the fields under a smoothed affinity $\bar{w}_j = 0.9\bar{w}_{j-1} + 0.1 w_j$:

$$\Omega^{(k)}_{j,i} = \gamma \Omega^{(k)}_{j-1,i} + w_{j,k} \frac{v_{j,i}}{\max\big(\text{mean}_{\ell \in P(i)}\, v_{j,\ell}, 10^{-30}\big)}, \quad \Omega^{\text{read}}_{j,i} = \sum_{k=1}^{K} \bar{w}_{j,k} \Omega^{(k)}_{j,i}. \tag{15}$$

For $K = 1$ this reduces to Eq. equation 14. The oracle variant of Table 2 replaces the routing affinity with ground-truth feedback labels.