BALANCE: BENCHMARKING CONTINUAL LEARNING UNDER CHANGING DEPLOYMENT BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learners adapt to changing tasks under deployment budgets that can vary independently. We introduce Benchmark for Adaptive Lifelonglearning with Network Control and Elasticity (BALANCE) for studying model lifecycle control under these joint changes. External requests reclaim capacity at announced deadlines and return it later. A controller schedules resizing, candidate training and deployment while the current learner serves the task stream. Preparation and updates consume interaction time; ready deployment is instantaneous. Two recurrent prediction families combine locally shared task structure with limited deployment capacity: shared linear rules support analytical capacity calibration, and shared nonlinear features support evaluation with ordinary task-conditioned MLPs. We develop a resource- and deadline-aware model-maintenance algorithm whose rule-based implementations provide reference baselines in BALANCE. An accompanying feasibility study uses imitation-initialized reinforcement learning to learn lifecycle decisions under the same resource protocol. Paired experiments show that maintenance cadence changes the trade-off between prediction performance and model-maintenance work at similar average deployed sizes. A controlled throughput-shift study in the linear family finds that sensitivity to maintenance speed accompanies increased candidate cancellation and reduced deployed capacity despite resource compliance.

1 INTRODUCTION

Continual learners may have to share their deployment platform. For example, a platform hosting several agents can reallocate resources as task priorities change; on an edge device, a foreground application can temporarily reduce the capacity available to a learning service. Dynamic resource–accuracy allocation for concurrent mobile applications is already a practical systems concern (Fang et al., 2018). These scenarios motivate studying how a learner responds to allocation decisions made by its host platform.

For a learner facing changing tasks, recovering capacity does not recover the knowledge removed by compression. The system must decide which model to maintain, when to prepare its replacement, and how much learning to perform before deployment. These decisions couple the adaptation of the learner to the timing of resource availability.

Compute-budgeted and real-time evaluations account for the cost of learning (Prabhu et al., 2023; Ghunaim et al., 2023). Expandable networks and reinforcement-learning-based structure selection trade capacity against task performance (Yoon et al., 2018; Xu & Zhu, 2018). We study the additional scheduling problem created when *an external process reclaims deployed capacity at announced future boundaries while model maintenance consumes interaction time*.

BALANCE separates the external resource scheduler, lifecycle controller and inner task learner. The environment advances resource allocations and audits compliance; the controller schedules maintenance; and the inner learner predicts and learns from the task stream. This separation gives different control algorithms a common interface and evaluation protocol.

We study a capacity-limited continual-learning setting: the deployed model is assumed unable to represent all task rules simultaneously, although local groups of related tasks are learnable. Two recurrent families instantiate this setting. Shared linear rules give an exact fixed-model capacity

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.

certificate for a bilinear learner; shared nonlinear features support empirical calibration with an ordinary task-conditioned MLP through local learning, transfer, revisitation and joint fitting.

An accompanying feasibility study uses imitation-initialized reinforcement learning (RL) to examine learned lifecycle decisions, maintenance work and sensitivity to resource changes.

Our contributions are threefold:

1. **An idealized mathematical formulation.** We formulate model lifecycle control in continual learning under exogenous, time-varying deployment-capacity constraints, with lifetime reward, per-boundary feasibility and timed model maintenance.
2. **The BALANCE benchmark.** We introduce a configurable framework with two continuing task families. A common maintenance-action interface, capacity calibration and trajectory audits support reproducible research on model resizing, learning schedules and continual adaptation.
3. **A resource- and deadline-aware maintenance algorithm.** We develop an algorithm coordinating model resizing, candidate training and deployment. The rule-based implementation in BALANCE provides reusable reference baselines through ASAP, ALAP and Periodic-ASAP scheduling.

2 RELATED WORK

Compute-aware continual evaluation. Ghunaim et al. (2023) evaluate online learning when data arrival proceeds during training. Prabhu et al. (2023) compare continual-learning methods under training-computation budgets and emphasize strong inexpensive baselines. These studies motivate explicit accounting for update cost. BALANCE couples a simulated maintenance service with an externally changing deployment ceiling within each lifetime, allowing the timing of learning and capacity adjustment to be studied together.

Adaptive architecture and bounded capacity. DEN selects and expands capacity during sequential learning (Yoon et al., 2018); Reinforced Continual Learning uses an RL controller for per-task architecture expansion (Xu & Zhu, 2018); and Learn to Grow separates structure selection from parameter learning (Li et al., 2019). NORACL triggers growth from representational and plasticity limitations (Raghunathan et al., 2026). PackNet reuses pruned capacity (Mallya & Lazebnik, 2018), while Progress & Compress transfers knowledge into a fixed-capacity knowledge base (Schwarz et al., 2018). These mechanisms address how to retain and allocate knowledge. BALANCE adds external deadlines and maintenance timing to the evaluation of such decisions.

Time-dependent resource formulations. ABAS jointly selects architectures and allocates resources through mixed-integer optimization, including time-dependent budgets (Liu et al., 2025). GRAPA combines gradient-guided reuse of frozen weights with A2C-based pruning under a fixed total parameter budget (Chen et al., 2026). BALANCE focuses on the learner's response to external reclamation notices, effective deadlines and matched resource returns. Candidate preparation and updates take place while the deployed learner continues serving tasks. This lifecycle protocol makes replacement timing and deadline compliance explicit parts of the learning problem, alongside capacity allocation.

Runtime elasticity and shared deployment. Slimmable networks support runtime width changes after shared-weight training (Yu et al., 2019). NestDNN allocates resources among concurrent mobile applications using multiple resource–accuracy operating points (Fang et al., 2018). These systems motivate our deployment setting. We study the continual learner's response to its allocation, including the learning needed to regain performance after capacity changes.

Shared-representation theory. Linear representation learning can improve new-task sample efficiency under explicit statistical assumptions (Tripuraneni et al., 2021). This provides a theoretical motivation for constructing related tasks from shared group features.

3 PROBLEM FORMULATION AND DEPLOYMENT PROTOCOL

3.1 EXOGENOUS CAPACITY AND TRAJECTORY FEASIBILITY

Parameter count provides a simple, hardware-independent deployment constraint. At fixed storage precision it is proportional to weight memory, making it a useful abstraction of deployment-memory contention. We account separately for the time spent preparing and updating models.

Let H count task interactions and t index their boundaries. The deployed learner M_t predicts; an outer controller uses public observations o_t to maintain it, preparing at most one candidate $\widetilde{M}_t$. For model M, let $\Theta_{\mathrm{unique}}(M)$ be its distinct parameter tensors, ϑ one such tensor, and numel its scalar-entry count. Parameter count $P(M)$ and deployed usage U_t are

$$U_t = P(M_t) = \sum_{\vartheta \in \Theta_{\mathrm{unique}}(M_t)} \mathrm{numel}(\vartheta). \tag{1}$$

All deployed heads, embeddings and biases count; shared tensors count once. Candidates are prepared on an idealized auxiliary node. Candidate parameters, replay, optimizer states and the outer controller lie outside this deployment budget.

Let K be the platform pool and $\ell_{\min}$ the minimum notice. External job j announces demand B_j at i_j, becomes effective at d_j with $d_j \geq i_j + \ell_{\min}$, and returns that allocation at r_j. Using $\mathbf{1}\{\cdot\}$ for an indicator, define external occupancy E_t and available deployment capacity C_t:

$$C_t = K - E_t = K - \sum_j B_j \mathbf{1}\{d_j \leq t < r_j\}. \tag{2}$$

Overlapping requests take effect without controller approval. Returns must match outstanding allocations; unmatched, duplicate or premature returns are invalid. Only issued requests and occurred returns are public.

Feasibility requires $U_t \leq C_t$ at boundaries zero through $H-1$, including post-maintenance checks; H has no interaction. Violations persist even if training continues; invalid actions become penalized no-ops. For trajectory τ, let $N(\tau)$ count completed interactions and $\mathrm{pass}(\tau)$ indicate eligibility:

$$\mathrm{pass}(\tau) = \mathbf{1}\{N(\tau) = H\}\, \mathbf{1}\{\text{complete audit}\}\, \mathbf{1}\{\text{valid actions}\}\, \mathbf{1}\{\text{no resource violation}\}. \tag{3}$$

We report the fraction of evaluated lifetimes satisfying this criterion.

3.2 ACTIONS, INTERACTION TIME, AND REWARD

Action a_t waits, cancels or deploys a candidate, or starts one with target width w, update budget n and automatic/manual deployment flag b: $a_t \in \{\textsc{Noop}, \textsc{Cancel}, \textsc{Deploy}, \textsc{Start}(w, n, b)\}$. Widths follow the supported integer topology.

One step is a prediction/label interaction, ordered as audit, maintenance, prediction/feedback, auxiliary work, ready automatic deployment, then the next audit. Ready manual deployments serve the current interaction; automatic ones serve the next. Neither erases earlier violations.

Let W_{prep} and W_{update} be declared preparation and per-update work for the chosen model/recipe, W_{job} the requested total, and ν the work serviced per interaction. Then

$$W_{\mathrm{job}} = W_{\mathrm{prep}} + nW_{\mathrm{update}}. \tag{4}$$

Preparation precedes updates. A job requires at least $\lceil W_{\mathrm{job}}/\nu \rceil$ interactions; waiting for feedback can extend its duration. Ready deployment is instantaneous, and cancellation retains the work already consumed. The service rate therefore determines how much learning can be completed while the task stream advances.

Let $\widehat{y}_t, y_t$ be scalar prediction and target, $e_t = \widehat{y}_t - y_t$ their error, and v_t, q_t resource-violation and invalid-action indicators with nonnegative penalties λ_v, λ_a. Operator clip projects its first argument onto the interval given by its last two. Task reward r_t^{task}, penalized reward r_t^{true} and lifetime score

$J_H(\tau)$ are

$$r_t^{\text{task}} = \text{clip}(1 - e_t^2, -1, 1), \tag{5}$$

$$r_t^{\text{true}} = r_t^{\text{task}} - \lambda_v v_t - \lambda_a q_t, \tag{6}$$

$$J_H(\tau) = \frac{1}{H} \sum_{t=0}^{H-1} r_t^{\text{true}}. \tag{7}$$

Maintenance policy π maximizes expected J_H subject to feasibility. Passing scores equal $1 - H^{-1} \sum_t \min(e_t^2, 2)$: one minus mean clipped squared error. Maintenance work is reported separately from this objective. We retain failed lifetimes and report reward and compliance.

4 The Balance benchmark

The two task families combine recurrent visits to related tasks with limited deployment capacity. Both use the interaction and maintenance protocol of Section 3.

4.1 Family I: shared linear rules and a capacity certificate

Partition N tasks into groups g with member sets $\mathcal{I}_g$. For input x of dimension d, group g has local rank r_g and hidden orthonormal-column basis $U_g \in \mathbb{R}^{d \times r_g}$. Member $i \in \mathcal{I}_g$ has an independent uniform unit-sphere coefficient $c_i \in \mathbb{R}^{r_g}$. Its fixed rule $w_i \in \mathbb{R}^d$ and scalar label y are, with $\top$ denoting transpose,

$$w_i = U_g c_i, \tag{8}$$

$$y = w_i^\top x. \tag{9}$$

Bases orthonormalize independent Gaussian matrices. Inputs have independent, equiprobable ± 1 coordinates, with zero mean and identity covariance. Each member defines a distinct prediction rule using its group's shared features.

Both families endlessly renew shuffled group cycles without consecutive group repeats. Visits have sampled dwell times; each step draws a uniform member and fresh input. The finite-bank generator does not depend on H. Task IDs are globally unique, randomly permuted and stable on revisits. Predictors receive the task ID and input (i, x) before predicting, followed by the target y. Group membership, switches and teacher parameters are hidden, so the learner adapts through observed inputs and feedback.

The deployed bilinear predictor has width r, task coefficients $A \in \mathbb{R}^{N \times r}$ and shared matrix $B \in \mathbb{R}^{r \times d}$. Write $A_{i,:}$ for row i, $\widehat{y}$ for prediction and $P(r)$ for the full count:

$$\widehat{y} = A_{i,:} B x, \tag{10}$$

$$P(r) = (N + d)r. \tag{11}$$

Let $W \in \mathbb{R}^{N \times d}$ stack the rule row vectors $w_i^\top$, $\sigma_j(W)$ be its singular values in descending order, and $\| \cdot \|_F$ the Frobenius norm. The fixed-model uniform-task population MSE satisfies

$$\min_{A,B} \frac{1}{N} \|W - AB\|_F^2 = \frac{1}{N} \sum_{j > r} \sigma_j(W)^2. \tag{12}$$

Maximum width $r_{\max}$ satisfying $\max_g r_g \le r_{\max} < \text{rank}(W)$ allows each group but forbids exact all-task representation by one fixed model. Thus the learner can represent the current group while adapting its shared representation across visits. The fixed-model result and the shared-feature arguments are derived in Appendix B.

Online learning and resizing. For learner parameters β, predictor M_β and a nonempty minibatch $\mathcal{B}$ of revealed triples (i, x, y), let $|\mathcal{B}|$ denote its size. Both base learners minimize the minibatch loss $L(\beta; \mathcal{B})$:

$$L(\beta; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(i,x,y) \in \mathcal{B}} (M_\beta(i, x) - y)^2. \tag{13}$$

Requested updates train the candidate while deployed weights remain fixed until replacement. Prefix shrinking retains overlapping parameters; growth preserves the current function by adding zero-output paths. Predictions use the deployed model, while stored feedback is reserved for candidate training.

4.2 Family II: shared nonlinear features

Here inputs have iid standard Gaussian coordinates. For feature dimension k, each group has an independent orthonormal-column basis $Q_g \in \mathbb{R}^{d\times k}$, scale $\alpha > 0$ and bias $b_g \in \mathbb{R}^k$. Vectors $\mu_g, s_g \in \mathbb{R}^k$ are the transformed coordinates' means and positive standard deviations. With coordinatewise $\tanh$ and division, shared features $\phi_g(x) \in \mathbb{R}^k$ and independent uniform unit-sphere member coefficients $v_i \in \mathbb{R}^k$ give

$$\phi_g(x) = \frac{\tanh(\alpha Q_g^\top x + b_g) - \mu_g}{s_g}, \tag{14}$$

$$y = v_i^\top \phi_g(x), \quad i \in \mathcal{I}_g. \tag{15}$$

Members combine the same nonlinear features with different coefficients. The task stream, information available to the learner and reward follow the same protocol as the linear family.

The learner is an ordinary task-conditioned two-hidden-layer MLP. Let e be embedding dimension, h hidden width, $E \in \mathbb{R}^{N\times e}$ the task embedding, and $[\cdot;\cdot]$ concatenation. Weights have shapes $W_1 \in \mathbb{R}^{h\times(d+e)}$, $W_2 \in \mathbb{R}^{h\times h}$; hidden states z_1, z_2, biases a_1, a_2 and readout w_o lie in $\mathbb{R}^h$, and a_o is a scalar bias. Its prediction and full count $P(h)$ are

$$z_1 = \tanh(W_1[x; E_{i,:}^\top] + a_1), \tag{16}$$

$$z_2 = \tanh(W_2 z_1 + a_2), \tag{17}$$

$$\widehat{y} = w_o^\top z_2 + a_o, \tag{18}$$

$$P(h) = Ne + h(d + e + 1) + h(h + 1) + h + 1. \tag{19}$$

All embeddings and biases contribute to the deployment count. We select the evaluation profile through local learning, same-group transfer, recurrent adaptation and joint-fitting experiments. This empirical calibration extends the capacity-limited design to nonlinear prediction.

5 Reference controllers

We develop a resource- and deadline-aware maintenance algorithm with three rule-based scheduling choices. A learned-control study uses the same public interface and deadline guard.

5.1 Resource- and deadline-aware maintenance

Inputs are public capacity notices, deployed/candidate status, a finite size menu, and maintenance costs and throughput. The output is a maintenance action; the inner learner supplies prediction, resizing and updating routines. Public observation o_t reports status, time and recent performance. Decisions are at most T_{dec} interactions apart; requests, capacity changes, readiness, the public clock or urgent recovery can trigger earlier decisions. Group switches are hidden.

Let $\mathcal{P}_t$ collect pending issued requests, $s \geq t$ be a future boundary, and $\overline{C}_t(s)$ capacity after known demands without assuming returns. For width menu $\mathcal{W}$ and parameter count $P(w)$, conservative target $w_t^\star$ is the largest width fitting all announced demands by the horizon:

$$\overline{C}_t(s) = C_t - \sum_{j\in\mathcal{P}_t:\, d_j\leq s} B_j, \tag{20}$$

$$w_t^\star = \max\{w \in \mathcal{W} : P(w) \leq \overline{C}_t(H)\}. \tag{21}$$

If no size fits, deadline recovery uses minimum width on a best-effort basis. For target width w and update budget n, nominal duration $\widehat{L}_t(w, n)$ is Eq. (4) divided by ν, rounded upwards. Updates may wait for feedback; zero-update copying has a known duration.

A controller-side guard reserves copying time. Let $\mathcal{A}_t$ be its admissible menu, d the first threatening deadline and k_0 a feasible copy's duration. Idle recovery starts by slack $k_0 + 1$; an obstructing candidate with uncertain completion is canceled by slack $k_0 + 2$. The target covers announced demands through the longest copy window after d. Recovery requires a fitting size and sufficient notice for cancellation, copying and deployment. Controllers share the guard; the environment independently advances capacity and audits execution.

Decision procedure. At each decision, refresh the public inputs and guarded menu, then execute the first applicable branch:

1. Execute a guard-prescribed recovery action, if present.
2. Deploy a ready candidate when deployment is admissible.
3. If a candidate remains, cancel it when ready and cancellation is admissible; otherwise wait.
4. When idle, apply the scheduling rule to request an admissible start or wait. Repeat after further task interactions.

5.2 Rule controller family

Three scheduling rules instantiate this procedure in Balance. Let n_0 be the fixed ordinary update budget.

ASAP. When idle, request $\textsc{Start}(w_t^\star, n_0, \mathrm{manual})$ if admissible, otherwise no-op. Thus announced demands affect the target immediately.

ALAP. When idle, initially target the largest currently fitting width. Let d_t, if any, be its first threatening announced deadline and w_t^- the largest width fitting $\overline{C}_t(d_t)$. For a defined feasible w_t^-, the target changes when

$$d_t - t \le T_{\mathrm{dec}} + \widehat{L}_t(w_t^-, n_0) + 1. \tag{22}$$

The extra step reserves pre-audit deployment; ALAP retains a larger model until this estimated preparation window.

Periodic-ASAP. Check size with period T_{size} and phase φ, and train at unchanged width on period T_{train}. Let w_t be current width, Δ_P the largest parameter increment between adjacent widths in the dense integer topology, and $\rho > 1$ a growth multiplier. Size checks shrink to $w_t^\star$ if needed; growth to that target requires

$$\overline{C}_t(H) - P(w_t) \ge \max\{\Delta_P + 1, \lceil \rho \Delta_P \rceil\}. \tag{23}$$

Otherwise width is retained. Shrinking can use a zero-update copy; the deadline guard takes priority over both clocks. As an auxiliary reference, Minimum-u ordinarily requests the smallest width with fixed update count u under the same contract.

5.3 Reinforcement-learning feasibility study

Using ASAP as an imitation teacher, we study learned scheduling over the same guarded menu. Decision $k \ge 0$ occurs at boundary t_k. A gated recurrent unit (GRU) encodes public-observation history in state $\mathbf{h}_k$. Let θ be outer actor–critic parameters, $\psi(o, a)$ the size, update-budget and duration descriptor of action a under observation o, and s_θ its score. Policy π_θ normalizes over admissible alternatives a'; current descriptors and menu are implicit in its conditioning:

$$\pi_\theta(a \mid \mathbf{h}_k) = \frac{\exp s_\theta(\mathbf{h}_k, \psi(o_{t_k}, a))}{\sum_{a' \in \mathcal{A}_{t_k}} \exp s_\theta(\mathbf{h}_k, \psi(o_{t_k}, a'))}, \quad a \in \mathcal{A}_{t_k}. \tag{24}$$

Training proceeds from behavioral cloning (BC) to DAgger and then PPO. BC fits teacher labels; DAgger (Ross et al., 2011) adds labels on student-visited histories, and PPO (Schulman et al., 2017) uses fresh complete lifetimes. For decision duration Δ_k, the centered macro reward R_k sums all interaction rewards until the next decision:

$$R_k = \sum_{t=t_k}^{t_k + \Delta_k - 1} (r_t^{\mathrm{true}} - 1). \tag{25}$$

Table 1: Nonlinear calibration on three held-out banks. Local: diagnostic passes/24; transfer: same-versus-cross-group target-MSE reduction. Revisit: three full constant-capacity lifetimes per bank. Joint: final 65,536-update MSE ranges (width 16 Adam/SGD; width 128 Adam).

Bank	Local	Transfer gain	Revisit reward	Joint MSE, $h = 16$	Joint MSE, $h = 128$
21	24/24	54.3%	0.9345	0.255–0.298	0.015–0.018
22	24/24	55.4%	0.9459	0.248–0.291	0.019–0.022
23	23/24	56.9%	0.9384	0.261–0.314	0.013–0.023

Centering subtracts the policy-independent constant H from lifetime return. Validation selects compliant checkpoints at each stage. Testing freezes controller parameters and uses greedy decisions with the common deadline guard; recurrent state and inner learning continue evolving. Training objectives appear in Appendix A.6.

6 Benchmark characterization

6.1 Evaluation setup

Both families contain eight groups of eight tasks. Linear/nonlinear profiles use widths $\{1, 2, 4, 8\}/\{8, 16\}$, pools of 768/881 parameters and 65,536/786,432 interactions per lifetime.

Methods share task/resource streams, observations, action menus and the guard. Rules use ten paired scenarios; learned methods use five linear or three nonlinear training seeds (five for Double DQN). Test controllers are frozen while inner learners adapt. Metrics are lifetime mean reward, mean deployed parameters and consumed work, including cancellations. Training/statistical settings and inner-recipe checks appear in Appendix A and Table 10.

6.2 Task structure and capacity calibration

Across 100 linear task banks, global rule rank is 32 while group ranks are at most eight. For the rank-eight bilinear learner, Eq. (12) gives mean fixed-model all-task MSE .326091, with 768 deployed parameters versus 3072 at rank 32. This separates local representation from joint capacity.

Table 1 summarizes three held-out nonlinear banks. At width 16, 71/72 fixed-group runs meet mean/worst-member MSE criteria of .05/.10. Same-group transfer and recurrent rewards above .93 support shared structure and adaptation. Joint fitting retains MSE .248–.314 at width 16, versus .013–.023 at width 128. This provides empirical capacity calibration for the selected MLP and training budgets.

6.3 Rule-based reference performance

The rule-reference panels of Tables 2 and 3 show that ASAP improves over Minimum by .35984 reward in the linear family and .09304 in the nonlinear family. These complete-policy gains combine capacity choice and maintenance. ALAP achieves similar observed means to ASAP, with a nonlinear difference of .00083 (95% interval $[-.00038, .00215]$).

6.4 Maintenance cadence and response timing

Periodic-ASAP reduces maintenance work by 22.68% in the linear family and 47.3% in the nonlinear family relative to ASAP, with reward decreases of .00555 and .01747. Average deployed size remains close to ASAP's: about 628 parameters in the linear family and 820 in the nonlinear family. Thus maintenance cadence offers a performance–work trade-off even at similar average model sizes; work is an auxiliary evaluation axis. A complementary 16-interaction minimum gap between voluntary starts changes the same trade-off; effects and uncertainty appear in Appendix C.

In 120 linear lifetimes with fixed effective capacity and task streams, the deferred-minus-early reward difference grows from .00029 to .00161 as notice increases from 64 to 256 at fast service; at

Table 2: Linear rule references and learned-control feasibility study ($H = 65{,}536$). All 240 lifetimes pass. n: test lifetimes; $\bar{P}$: mean deployed parameters. Work: millions of declared units.

Controller	n	Reward	$\bar{P}$	Update	Prepare
A. Rule-based references					
Minimum	10	0.4067	96.0	100.62	2.10
ASAP	10	0.7665	627.8	159.63	3.33
ALAP	10	0.7677	628.9	159.73	3.33
Periodic-ASAP	10	0.7609	627.5	123.43	2.57
B. Learned-control feasibility study					
BC	50	0.7016	516.6	149.67	3.15
DAgger	50	0.7606	629.0	161.07	3.38
Post-PPO	50	0.7604	625.0	160.78	3.38
Scratch PPO	50	0.2741	481.5	77.73	4.79

Table 3: Nonlinear rule references and learned-control feasibility study ($H = 786{,}432$). All 150 lifetimes pass. n: test lifetimes. Other columns: lifetime means; work in billions of declared units.

Controller	n	Reward	Parameters	Work (G)	Guard
A. Rule-based references					
Minimum	10	0.8351	505.0	41.68	0.0
ASAP	10	0.9281	820.1	46.60	0.0
ALAP	10	0.9290	820.2	46.60	0.6
Periodic-ASAP	10	0.9107	820.1	24.54	2.6
B. Learned-control feasibility study					
DAgger	30	0.9255	819.0	46.58	5.7
Post-PPO	30	0.9217	820.1	46.62	5.6
Double DQN	50	0.4101	519.5	3.49	0.6

slow service and notice 256 it is .00201 (Table 9). In these profiles, cadence changes work substantially, while early versus deferred shrinking has a small effect on lifetime reward. Event-window analysis also captures localized responses to notice changes (Figure 2).

6.5 Feasibility of learned lifecycle decisions

Teacher-initialized DAgger reaches .7606/.9255 reward in the linear/nonlinear families; post-PPO reaches .7604/.9217 with the common guard (Figure 1). PPO fine-tuning produces slightly lower mean rewards than DAgger. Linear scratch PPO and nonlinear Double DQN obtain .2741/.4101, illustrating sensitivity to training.

Sensitivity to maintenance speed. Table 4 evaluates frozen-policy transfer on paired linear streams. At default capacity, halving throughput reduces DAgger/post-PPO reward by .33492/.42116; ASAP decreases by .02332. Deeper reclamation at fast throughput causes smaller learned-policy losses of .02485/.02222. These shifts identify maintenance speed as a key transfer condition.

Post-hoc analysis reveals a behavioral change (Table 8). At default capacity, canceled candidates' share of completed-update work rises from 3.0% to 59.0% for DAgger and from 2.8% to 72.1% for post-PPO; ASAP's share remains zero. Under slow service, update-work totals are similar (89.16/87.88/86.26 million for ASAP/DAgger/post-PPO), but mean deployed parameters fall to 331.5/291.7 for the learned policies, versus 651.2 for ASAP. The learned policies cancel 2290/2744 candidates per lifetime, versus only 3.58/3.64 forced-guard decisions. Most cancellation therefore occurs outside forced recovery. Comparable consumed work yields different deployment outcomes; the behavioral association motivates testing cancellation decisions directly.

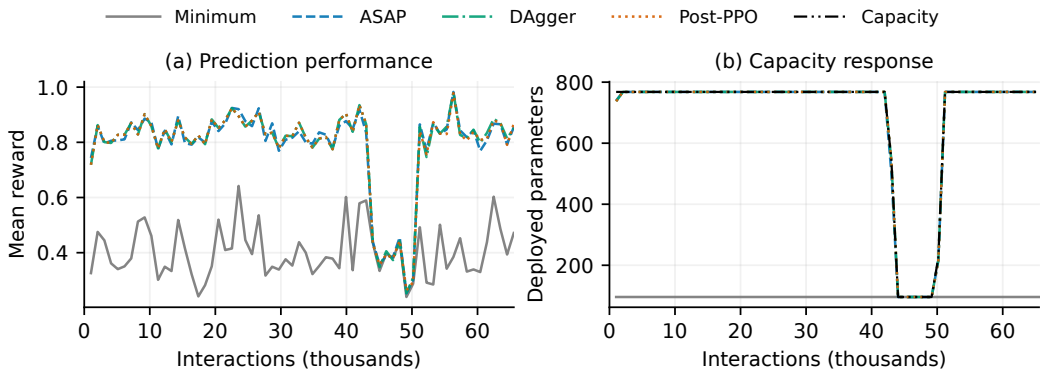

Figure 1: Linear main-test scenario zero (learned-controller seed 100): means over disjoint 1024-interaction blocks show (a) prediction reward and (b) deployed parameters and available capacity.

Table 4: Resource-shift sensitivity: mean reward for rule references and frozen learned policies. All 520 lifetimes pass; dwell/notice are 1024/64. Fast/slow maintenance supplies 3072/1536 work units per interaction.

	Fast maintenance		Slow maintenance	
Controller	Default	Deeper	Default	Deeper
A. Rule-based references				
Minimum	0.4100	0.4100	0.4103	0.4103
ASAP	0.7810	0.7508	0.7577	0.7293
ALAP	0.7820	0.7523	0.7574	0.7296
B. Frozen learned policies				
DAgger	0.7724	0.7475	0.4374	0.4337
Post-PPO	0.7698	0.7476	0.3487	0.3399

All 520 factorial lifetimes pass with the guard. A companion 16,384-interaction, 32-unit-GRU ablation causes 30/30 failures per learned stage when the guard is removed. Guarded feasibility and effective predictive adaptation are distinct outcomes.

7 Discussion and Conclusion

BALANCE formalizes continual learning under changing deployment budgets, with two calibrated task families and a rule-based maintenance algorithm. In the evaluated profiles, maintenance cadence changes the performance–work trade-off at similar average sizes; early versus deferred shrinking gives small lifetime gains. Under linear throughput shift, learned policies consume similar update work yet discard more candidates and deploy less capacity.

Evaluation uses finite recurrent banks, task-ID-aware inner learners, family-specific capacity calibration and three nonlinear training seeds. The auxiliary-node abstraction separates deployed capacity from candidate-training resources.

Future work. We plan to open-source the environments, rule and learned controllers, and reproduction tools upon acceptance. Research directions include notice-to-maintenance time ratios, cancellation control with deadline recovery, cross-rate training, reinitialization for plasticity (Sokar et al., 2023; Dohare et al., 2024), and extending the auxiliary-node model to shared training resources, candidate/optimizer memory and transfer, loading and deployment delays.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, retrieve and discover relevant literature, draft sections of the paper, and assist with parts of the code implementation, debugging and testing. The authors take responsibility for the final content of this work, including AI-assisted text, claims and code.

ETHICS STATEMENT

The study uses synthetic prediction tasks and simulated resource schedules. Its resource and feasibility measures describe the benchmark's mathematical contract.

REPRODUCIBILITY STATEMENT

We provide an anonymized source-code package in the supplementary material for reviewer reproduction. It includes the resource protocol, both task families, rule-based and learned controllers, configurations, training/evaluation scripts and tests. Its README and reproduction documentation explain installation, execution and code–paper correspondence. The package supports fresh training and evaluation; historical checkpoints and raw trajectories are separate archived inputs, outside this source-only submission. The appendix specifies experimental settings and statistical procedures. The correspondence documentation distinguishes migrated examples from historical result configurations.

REFERENCES

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.05641.

Yitian Chen, Shigeng Zhang, Xuan Liu, Mingming Lu, Kai Chen, Hongye Zhu, and Xinning Chen. Parameter-efficient continual learning for enhancing plasticity without forgetting under limited model capacity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10789–10798, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Chen_Parameter-efficient_Continual_Learning_for_Enhancing_Plasticity_without_Forgetting_under_Limited_CVPR_2026_paper.html.

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. doi: 10.1038/s41586-024-07711-7. URL https://www.nature.com/articles/s41586-024-07711-7.

Biyi Fang, Xiao Zeng, and Mi Zhang. NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 115–127, 2018. doi: 10.1145/3241539.3241559. URL https://arxiv.org/abs/1810.10090.

Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip H. S. Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new hope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11888–11897, 2023. URL https://arxiv.org/abs/2302.01047.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3925–3934, 2019. URL https://proceedings.mlr.press/v97/li19m.html.

Weijun Liu, Haoming Chen, Leifang Zhan, Jiaxin Wu, and Zhou Shen. Adaptive bottleneck architecture search for resource-constrained continual learning. Research Square preprint, version 1,

2025. URL https://doi.org/10.21203/rs.3.rs-8201329/v1. Posted November 26, 2025; not peer reviewed. Author names follow repository cover metadata.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Mallya_PackNet_Adding_Multiple_CVPR_2018_paper.html.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip H. S. Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3698–3707, 2023. URL https://arxiv.org/abs/2303.11165.

Karthik Charan Raghunathan, Christian Metzner, Laura Kriener, and Melika Payvand. NORACL: Neurogenesis for oracle-free resource-adaptive continual learning. *arXiv preprint arXiv:2604.27031*, 2026. URL https://arxiv.org/abs/2604.27031.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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 *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4528–4537, 2018. URL https://proceedings.mlr.press/v80/schwarz18a.html.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32145–32168, 2023. URL https://proceedings.mlr.press/v202/sokar23a.html.

Gábor J. Székely and Maria L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013. doi: 10.1016/j.jspi.2013.03.018. URL https://doi.org/10.1016/j.jspi.2013.03.018.

Nilesh Tripuraneni, Chi Jin, and Michael I. Jordan. Provable meta-learning of linear representations. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10434–10443, 2021. URL https://proceedings.mlr.press/v139/tripuraneni21a.html.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. URL https://arxiv.org/abs/1904.07734.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pp. 2094–2100, 2016. doi: 10.1609/aaai.v30i1.10295. URL https://arxiv.org/abs/1509.06461.

Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper/2018/hash/cee631121c2ec9232f3a2f028ad5c89b-Abstract.html.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1708.01547.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1812.08928.

A EXPERIMENTAL SETTINGS

A.1 TASK STREAMS AND INNER LEARNERS

Both main profiles contain eight groups of eight tasks. The linear profile uses input dimension 32, group ranks $(2, 4, 8, 2, 4, 8, 4, 8)$, widths $\{1, 2, 4, 8\}$ and a deployment pool of 768 parameters. It starts at width one. The nonlinear profile uses input dimension 16, four teacher features per group, feature scale two, task-embedding dimension four and widths $\{8, 16\}$. It starts at width eight and has a pool of 881 parameters. Teacher biases are uniform on $[-.5, .5]$; feature normalization uses Gaussian quadrature of order 192, checked against order 256 at tolerance 10^{-6}.

Group visits have uniformly sampled integer durations within $\pm 25\%$ of 1024 interactions in the linear profile and 32,768 in the nonlinear profile. Lifetimes contain 65,536 and 786,432 interactions, respectively. External requests are sampled independently of the task stream, with minimum notice 64, matched returns and feasible capacity schedules.

Both inner learners use Adam, minibatches of 32, a FIFO replay capacity of 256 and gradient clipping at five. Their learning rates are .01 and .001. A minibatch becomes available after 32 labels; examples remain reusable until eviction. The linear implementation additionally exposes a recent-feedback view of 32 examples, giving at most 288 distinct retained examples. Resizing copies overlapping parameters and optimizer moments. Growth initializes new features with zero read-out contribution, preserving predictions (Chen et al., 2016); prefix shrinking retains the overlapping coordinates.

A.2 RESOURCE-REQUEST GENERATOR

The main resource process uses four independently sampled latent capacity intervals. Let $\ell = 64$ be the notice lead, $L = H - \ell$ the remaining calendar length, and $a = \max(1, \lfloor L/30 \rfloor)$ and $b = \max(1, \lfloor L/8 \rfloor)$ its duration limits. Each interval starts at an integer sampled uniformly from $\{\ell, \dots, H-a\}$. Conditional on start s, its duration is uniform on $\{a, \dots, \min(b, H-s)\}$; its capacity is uniform over the supported parameter counts (96/192/384/768 for the linear profile, 505/881 for the nonlinear profile). Effective capacity is the pointwise minimum of active interval caps and the full pool. Selecting the full pool can therefore produce no reduction.

The generator then decomposes this envelope into additive external jobs. A downward capacity change opens demand equal to the drop. An upward change returns the most recently opened outstanding amounts first, splitting an allocation into jobs with common issue/effective times when different portions return at different boundaries. Each job is announced exactly ℓ interactions before becoming effective and has one matching return. These decomposed jobs may overlap; the four sampled intervals are not four independent additive demands. Controllers observe only issued jobs and returns that have occurred. Task and resource random streams are separate. This construction reproduces the full effective-capacity trace while preserving the protocol's information restriction.

A.3 RESOURCE AND MAINTENANCE ACCOUNTING

All deployed parameter tensors are charged, including embeddings, heads and biases. Dense masked tensors retain their allocated size. Candidate preparation, feedback storage and optimizer state use the auxiliary training node. Each prediction is evaluated before its label becomes available for training.

For target parameter count P, the linear profile charges $W_{\text{prep}} = P$ and $W_{\text{update}} = 3P$, with service rate $\nu = 3072$. For current count P_c, the nonlinear profile charges

$$W_{\text{prep}} = 3P + 3\min(P_c, P), \tag{26}$$

$$W_{\text{update}} = 3 \cdot 32(P - Ne + e) + 4P, \tag{27}$$

with $\nu = 65{,}536$. The latter accounts for initialization, copying, optimizer buffers, a minibatch forward/backward pass and an optimizer update. These declared work units define the simulated service; hardware-independent FLOP equivalence is outside this accounting. Ready deployment has $W_{\text{switch}} = 0$. Resource-violation and invalid-action penalties are $\lambda_v = 100$ and $\lambda_a = 1$.

For context, a 16-update job at maximum width needs at least 13 linear or 16 nonlinear interactions under these work rates; nonlinear shrinking from width 16 to 8 needs at least seven. The default notice is 64 interactions. These work-only durations exclude feedback waits. Changing the ratio between notice, job duration and group dwell changes the scheduling conditions.

The boundary audit occurs before manual replacement. A ready manual deployment serves the current interaction, whereas automatic completion serves the next. Canceled and partial work remain charged. Candidate filtering restricts targets to the permanent pool; the deadline guard additionally handles announced transient reductions.

A.4 Public controller observations

Table 5 lists the complete input projection. Here T_{clk} is the public clock period (1024 or 32,768), $w_{\max}$ and $P_{\max}$ are maximum menu width and parameter count, and the other quantities follow Section 3. The clock phase follows the public interaction clock, not the hidden, jittered group-visit phase. Pending requests have been issued but have not yet become effective. An absent next request has normalized delay one and amount zero; an absent threatening deadline has normalized delay one. Missing candidate fields and pre-feedback task summaries are zero. At reset, performance and decision-history summaries are zero; time since deployment uses boundary zero as its reference. The final eight slots contain four regression summaries and four padding zeros, not an eight-way task identifier.

Table 5: The 38-dimensional public controller projection, in implementation order. Task IDs, group switches, teacher parameters and future returns are excluded.

Indices	Features and normalization
1–3	Elapsed fraction t/H, public phase $(t \bmod T_{\mathrm{clk}})/T_{\mathrm{clk}}$, and remaining fraction $(H-t)/H$.
4–7	Available capacity C_t/K, deployed usage U_t/K, headroom $(C_t - U_t)/K$, and external occupancy $(K - C_t)/K$.
8–9	Deployed width divided by $w_{\max}$; deployed parameters divided by $P_{\max}$.
10–14	Candidate-exists flag, width/$w_{\max}$, fraction of job work consumed, ready flag, and parameters/$P_{\max}$.
15–17	Remaining and total candidate work, each divided by νT_{clk}; flag indicating whether preparation or a feedback-supported update can advance.
18–20	Reward EMA (coefficient .05), last penalized reward, and mean reward over the preceding controller decision interval.
21–26	Pending request count/8; pending demand/K; next effective-time delay/H and amount/K; first threatening deadline delay/H; conservative future capacity $\overline{C}_t(H)/K$.
27–30	Deployment-at-current-boundary flag; interactions since deployment/T_{clk}; last action code/3 (no-op, cancel, deploy, start: 0–3); last requested update count/64.
31–38	Last observed squared prediction error; one minus that error (unclipped); deployed learner's last minibatch loss; replay fill fraction; four zero padding entries.

Action descriptors contain four kind indicators, normalized width, parameter count, resource usage, requested updates, nominal duration, signed width change, future headroom, deadline slack, and same-width/zero-update flags. Counts use the corresponding menu maxima, resource quantities use K, duration uses T_{clk}, and slack uses H. Slack subtracts nominal duration and one deployment step; its normalized value is one when no deadline threatens. Width and update one-hots are appended. The guarded action menu and descriptors use the same public information.

A.5 Controllers and training

All controllers use decision interval $T_{\mathrm{dec}} = 16$, update choices $\{0, 1, 4, 16, 64\}$ and manual deployment. Ordinary rule candidates request $n_0 = 16$ updates. Periodic-ASAP uses $T_{\mathrm{size}} = 64$, $\varphi = 0$, $T_{\mathrm{train}} = 16$ and growth multiplier $\rho = 1.5$. The archived identifiers Latest-feasible and ASAP-ver2 denote ALAP and Periodic-ASAP, respectively.

The learned controller has 38 public observation features, a 64-unit GRU and action descriptors with 14 numeric features plus width/update one-hots. The outer encoder uses performance summaries rather than task IDs. BC weights balance action-kind, direction and target classes using training labels, with inverse-frequency weights capped at four and normalized per lifetime. Three teacher lifetimes supply 120 BC optimizer steps. Two DAgger rounds each collect three student lifetimes and perform 60 supervised steps. A 40-step value-only warm-up precedes eight PPO iterations, each with three fresh lifetimes and two optimization epochs. PPO uses learning rates $.0003/.001$ for actor/critic, reward scale .01, clipping tolerance .2, entropy coefficient .001, value coefficient .5 and gradient clipping .5. Value gradients update the critic head separately from the actor representation. The scratch-PPO reference uses actor learning rate .0001.

Validation evaluates three scenarios every two PPO iterations and selects compliant checkpoints separately for each reported stage. Test scenarios are held out, and frozen policies choose greedy actions without teacher queries. Linear learned methods use five training seeds. Nonlinear DAgger/post-PPO use seeds 100–102: a time-budget amendment after the first two seeds reduced the planned five-seed cohort to its first three seeds in the original order. Per-seed budgets and validation selection remained fixed; the interrupted fourth attempt is retained in the research records.

The nonlinear Double DQN reference (van Hasselt et al., 2016) scores actions with two 64-unit tanh layers. It uses the common observation, menu and guard, 50,000 replay entries, Adam(.0003), batch 128 and gradient clipping five. Updates begin after 4096 interactions and occur once per 256 interactions; target weights are copied every 500 optimizer steps. Exploration decreases from .30 to .05 over the first 80% of training. Its five seeds each receive 33 training and 24 validation lifetimes, matching the warm-start interaction budgets. The recurrent and feedforward architectures and optimization procedures differ.

A.6 Learning objectives

For decision index k and policy parameters θ from Section 5.3, let f_θ encode the public observation o_{t_k}. The GRU state $\mathbf{h}_k$ is initialized with $\mathbf{h}_{-1} = 0$ and updated by

$$\mathbf{h}_k = \mathrm{GRU}_\theta(\mathbf{h}_{k-1}, f_\theta(o_{t_k})). \tag{28}$$

For trajectory τ, let $\mathcal{D}_\tau$ collect decisions with multiple admissible actions, $|\mathcal{D}_\tau|$ its cardinality, a_k^{teach} the ASAP teacher action and $\omega_k > 0$ its weight. Behavioral-cloning loss L_{BC} is

$$L_{\mathrm{BC}}(\tau) = -\frac{1}{|\mathcal{D}_\tau|} \sum_{k \in \mathcal{D}_\tau} \omega_k \log \pi_\theta(a_k^{\mathrm{teach}} \mid \mathbf{h}_k). \tag{29}$$

Lifetimes with discretionary decisions contribute equally; training-label weights are uniform or class-balanced. DAgger repeats this loss on aggregated teacher labels from student-visited histories.

After value-only warm-up, PPO collects fresh complete lifetimes with decision count m_τ and executed actions a_k. Let G_k be scaled undiscounted return, $\eta > 0$ its scale and z_k the likelihood ratio. Rollout policy π_{old} and states $\mathbf{h}_k^{\mathrm{old}}$ are frozen; ℓ indexes remaining decisions and R_ℓ is defined by Eq. (25):

$$G_k = \eta \sum_{\ell=k}^{m_\tau - 1} R_\ell, \tag{30}$$

$$z_k = \frac{\pi_\theta(a_k \mid \mathbf{h}_k)}{\pi_{\mathrm{old}}(a_k \mid \mathbf{h}_k^{\mathrm{old}})}. \tag{31}$$

For frozen rollout critic V_{old}, standardize $G_k - V_{\mathrm{old}}(\mathbf{h}_k^{\mathrm{old}})$ over the batch's discretionary decisions to obtain advantage $\widehat{A}_k$. With clipping tolerance $\epsilon \in (0, 1)$, per-lifetime actor loss $L_{\mathrm{actor}}(\tau)$ is

$$L_{\mathrm{actor}}(\tau) = -\frac{1}{H} \sum_{k \in \mathcal{D}_\tau} \min\{z_k \widehat{A}_k, \mathrm{clip}(z_k, 1-\epsilon, 1+\epsilon)\widehat{A}_k\}. \tag{32}$$

Value MSE and entropy regularization complete the loss; states are recomputed from complete histories. Actor updates use discretionary decisions; returns include forced-recovery rewards.

A.7 Evaluation, uncertainty and cost

Each rule is evaluated on ten scenarios. Each learned seed is evaluated on those same scenarios, pairing task, resource and initialization streams. The statistical units are training seeds and scenarios. Linear primary contrasts use 10,000 paired bootstrap draws and 97.5% intervals for Bonferroni 95% family coverage. Nonlinear and supporting comparisons use 20,000 paired draws and exploratory 95% intervals; learned comparisons resample seeds and scenarios separately. Failed complete lifetimes remain in reward summaries.

We report realized online work separately from outer training. Linear warm training uses 3,735,552 interactions per seed including validation, versus 2,752,512 for scratch PPO (1.357 times), with 312 versus 32 outer optimizer steps. BC, DAgger and post-PPO are stages of the same training chains. Nonlinear warm training uses 25,952,256 training and 18,874,368 validation interactions per seed, with 312 optimizer steps; Double DQN uses 101,361 steps. Teacher queries contribute to warm-start development and validation, while all test policies operate independently. These budgets make the end-to-end policy comparison and its costs explicit.

Within each main comparison, all methods use the same CPU evaluation runtime. Nonlinear warm training uses CPU rollouts and CUDA outer optimization; Double DQN training uses CPU. Source, configuration, checkpoint and trajectory identities are recorded, and completed evaluations undergo replay and per-boundary feasibility checks. Missing historical hardware telemetry is left unavailable.

Reproduction materials. The anonymous source archive contains the protocol, both task families, rule and learned controllers, configurations, experiment entry points and tests. Its `docs/REPRODUCTION.md` specifies installation and paired evaluation; `docs/PAPER_RESULTS.md` distinguishes the cleaned implementation from the frozen historical source/runtime/scenario identities associated with reported scores. The source bundle supports rerunning the methods; historical checkpoints and raw trajectories are separate evidence artifacts.

B Task structure and capacity calibration

B.1 Linear representation and task identity

For covariance-identity inputs, a task's prediction MSE equals its squared coefficient error. Averaging uniformly over tasks gives $N^{-1}\|W - AB\|_F^2$. Since AB has rank at most r and every rank-r matrix admits such a factorization, truncated SVD yields Equation (12). Across the 100 evaluated banks, group ranks match their specification and the rank-eight MSE bound ranges from .286871 to .360870.

Shared features also induce a useful notion of task similarity. Let $U, V \in \mathbb{R}^{d\times r}$ have orthonormal columns and let c_1, c_2, c_3 be iid coefficients with finite first moment. Energy-distance nonnegativity (Székely & Rizzo, 2013) gives

$$\mathbb{E}\|Uc_1 - Uc_2\|_2 \le \mathbb{E}\|Uc_1 - Vc_3\|_2. \tag{33}$$

The two within-distribution distances both equal $\mathbb{E}\|c_1 - c_2\|_2$; their energy distance is twice the cross distance minus twice this value. Equality is possible when the induced distributions coincide.

For a group basis U_g, define projection $\Pi_g = U_g U_g^\top$, and let I_d denote the d-dimensional identity. For target coefficient vector w, learned readout a, and a unit-sphere coefficient c in a second group of rank r_h, the approximation errors are

$$\min_a \mathbb{E}_x[(a^\top U_g^\top x - w^\top x)^2] = \|(I_d - \Pi_g)w\|_2^2, \tag{34}$$

$$\mathbb{E}_c\|(I_d - \Pi_g)U_h c\|_2^2 = 1 - \frac{\|U_g^\top U_h\|_F^2}{r_h}. \tag{35}$$

A same-group representation exactly supports a new member's rule, while cross-group error depends on subspace overlap. Under the statistical assumptions of Tripuraneni et al. (2021, Theorem 4), an independently learned representation with subspace error δ gives new-task least-squares excess prediction risk of order $\widetilde{O}(\delta^2 + r/n)$ from n samples. This motivates reusing shared features across related tasks.

Task identity specifies which group member's prediction is requested (van de Ven & Tolias, 2019). For a group with m members, define its mean rule $\overline{w}_g = m^{-1} \sum_{i \in \mathcal{I}_g} w_i$. If only the group and input are observed, the best predictor f has risk

$$\min_f \mathbb{E}_{i,x}[(f(x,g) - w_i^\top x)^2 \mid g] = 1 - \|\overline{w}_g\|_2^2. \tag{36}$$

Its generating-distribution mean is $1 - 1/m$, or $7/8$ for eight members. Providing stable task IDs resolves this target ambiguity. The task-specific coefficients remain part of the charged deployment model.

B.2 Nonlinear calibration

The nonlinear profile was selected on development banks using local learning, same-group transfer, recurrence and balanced all-task fitting. Independent confirmation uses three fresh banks, seeds 601–603, and the frozen inner learning recipe. Each bank has 24 fixed-group lifetimes, 24 three-arm transfer tests, three recurrent lifetimes and nine joint fits. These diagnostics fix capacity at width 16 to characterize the task family before introducing dynamic resource requests.

The single local failure has mean MSE .0443 and worst-member MSE .1185, exceeding the .10 worst-member criterion. All nine recurrent lifetimes pass the diagnostic checks, covering 214 complete exits and 142 observed at-risk recoveries. Here an at-risk visit begins with a previously learned group above the error threshold, and recovery is an observed return below that threshold during the visit. Repeated visits are summarized within lifetimes.

Joint fitting uses 65,536 updates and batch 128, comparing width-16 Adam/SGD with width-128 Adam (Table 1).

C Supporting analyses

C.1 Uncertainty in reference and feasibility results

Linear primary reward contrasts use 97.5% intervals with Bonferroni 95% family coverage: ASAP minus Minimum is .35984 $[.34250, .37706]$; DAgger minus ASAP is $-.00588$ $[-.01704, .00133]$.

In the nonlinear family, DAgger/post-PPO minus ASAP are $-.00266/-.00642$, with exploratory crossed 95% intervals $[-.00783, .00192]/[-.02123, .00103]$. Post-PPO minus DAgger is $-.00376$, with interval $[-.01634, .00088]$. ALAP-minus-ASAP is .00083, with interval $[-.00038, .00215]$. Thus the measured differences among these methods are small relative to their uncertainty. The three-seed post-PPO cohort includes the lower-reward .8039 test case.

C.2 Maintenance frequency

We impose a minimum gap of 16 interactions between voluntary candidate starts, leaving deadline recovery unchanged. The four linear cells use ten paired scenarios each (Table 6). Minimum/ASAP work falls by 84.2%/56.3%, and reward by .00713/.02810. The ASAP-minus-Minimum gap narrows by .02096 (exploratory paired 95% interval $[-.02358, -.01813]$).

Table 6: Linear size-policy/rate intervention: 40 compliant lifetimes, ten paired scenarios, $H = 65{,}536$. Ready/Gap 16 denote original/rate-gated starts; forced guard actions are unchanged. Work includes preparation and training; the table reports each policy's realized updates and costs.

	Mean reward		Work (M)		Updates (k)	
Rule	Ready	Gap 16	Ready	Gap 16	Ready	Gap 16
Minimum	0.4060	0.3988	102.71	16.24	349.36	55.23
ASAP	0.7724	0.7443	165.13	72.09	111.90	38.15

For nonlinear Minimum/ASAP, the same intervention changes reward by $+.00911/-.00389$; the corresponding intervals are $[-.00693, .02290]/[-.00643, -.00186]$. The gated ASAP-minus-

Minimum gap remains .08004, with interval $[.06480, .09447]$. These experiments show how update timing contributes to complete-policy performance; width and realized work also vary between policies.

C.3 DEADLINE GUARD

Guard removal is evaluated with 16,384-interaction linear lifetimes and a 32-unit recurrent controller: ten scenarios per rule and three frozen training seeds per learned stage. It causes all 30 DAgger and all 30 post-PPO lifetimes, plus two of ten Periodic-ASAP lifetimes, to violate the contract. All 100 guarded counterparts pass. Table 7 includes failed lifetimes' penalized rewards, showing the guard's contribution to deadline handling.

Table 7: Paired guard-removal ablation, $H = 16{,}384$: ten scenarios per rule and three seeds by ten scenarios per learned stage. All 100 guarded lifetimes pass; the unguarded evaluations contain 62 failures, included in the penalized reward means.

Method	Guarded reward	Unguarded reward	Unguarded pass
Minimum	0.4059	0.4059	10/10
ASAP	0.7672	0.7672	10/10
Periodic-ASAP	0.7653	0.7618	8/10
ALAP	0.7717	0.7717	10/10
Strat. DAgger	0.7667	-19.7106	0/30
Norm. PPO	0.7654	-19.5587	0/30

C.4 RESOURCE-SHIFT DESIGN

The linear factorial pairs task/resource streams across four cells with dwell 1024, notice 64 and throughput 3072 or 1536. Deeper reclamation keeps full capacity 768 and maps lower capacity C to $\max(96, \lfloor C/2 \rfloor)$. Each cell uses ten scenarios per method, with five frozen training seeds per learned method, totaling 520 lifetimes. Table means average seeds before scenarios.

C.5 CANDIDATE-WORK DIAGNOSTICS UNDER THROUGHPUT SHIFT

Table 8 summarizes the accepted default-capacity factorial records. The cancellation share is the sum of completed-update work on subsequently canceled candidates divided by total completed-update work. Preparation and unfinished update credit are excluded from both totals; completed updates on terminal undeployed candidates remain in the denominator. The fraction measures discarded update work, not cancellation frequency. Means use the same paired scenarios and fixed training seeds as Table 4.

Table 8: Post-hoc diagnostics for the default-capacity columns of Table 4. $\bar{P}$: mean deployed parameters; Update: completed-update work (millions); Cancel: fraction of that work spent on subsequently canceled candidates; Deploy/Guard: mean deployment/forced-guard counts per lifetime. Each row uses ten scenarios, crossed with five training seeds for learned policies.

Controller	Service	$\bar{P}$	Update	Cancel	Deploy	Guard
ASAP	Fast	651.4	162.61	0.0%	6586	0.00
ASAP	Slow	651.2	89.16	0.0%	3730	0.00
DAgger	Fast	651.4	163.61	3.0%	6209	8.56
DAgger	Slow	331.5	87.88	59.0%	1850	3.58
Post-PPO	Fast	638.3	162.79	2.8%	6348	8.46
Post-PPO	Slow	291.7	86.26	72.1%	1923	3.64

With slow service, DAgger/post-PPO cancel 2290/2744 candidates per lifetime on average, while their forced-guard decision counts are only 3.58/3.64. Thus forced recovery cannot account for most cancellations. Deployments and capacity use fall for the learned policies, whereas ASAP remains stable. This post-hoc accounting identifies loss of candidate work before deployment; an intervention would be needed to isolate which decision features trigger that behavior.

C.6 Controlled notice-window comparison

The 120 linear lifetimes cross notice leads 64/128/256, service rates 3072/256 and early/deferred starts over ten paired scenarios. All use $H = 65{,}536$, initial rank one and 16-update ordinary jobs. Each scenario's effective resource calendar is generated once with lead 256; only issue times change between notice conditions. Task, input and effective capacity streams remain paired.

Early uses ASAP. The deferred variant uses the same conservative target but delays shrinking until the estimated shrink-job duration plus one decision interval and one deployment step remain. Before this window, it admits a current-width job only if that job can finish while preserving the shrink window; otherwise it waits. Both use the common guard. This controlled admission variant is distinct from the main-table ALAP implementation. Table 9 reports every condition. At fast service, increasing notice from 64 to 256 reduces early reward by .00132 while all ten deferred scores remain unchanged. Longer notice changes when the early rule compresses; the deferred rule retains its timing in this comparison.

Table 9: Notice/maintenance timing: all six paired comparisons, ten scenarios each (120 compliant lifetimes). Effective capacity and task streams are fixed across notice leads. Intervals are exploratory paired 95% intervals; realized model paths and work may differ.

ν	Notice	Early	Deferred	Difference	95% interval
3072	64	0.7849	0.7851	+0.0003	$[-0.0006, +0.0012]$
3072	128	0.7846	0.7851	+0.0006	$[-0.0008, +0.0019]$
3072	256	0.7835	0.7851	+0.0016	$[+0.0005, +0.0026]$
256	64	0.5534	0.5530	-0.0004	$[-0.0011, +0.0000]$
256	128	0.5524	0.5536	+0.0012	$[-0.0002, +0.0026]$
256	256	0.5519	0.5539	+0.0020	$[+0.0003, +0.0038]$

Event-centered response. Figure 2 compares notice 32 against 64 on a separate paired linear cohort: 280 lifetimes, ten scenario calendars and all 27 requests. Reward is averaged within the union of $[-64, +256)$ interaction windows around effective boundaries; overlapping interactions count once. We average training seeds within each scenario, then scenarios equally. Shorter notice changes ASAP's event-window reward by +.03085 and lifetime reward by +.00150; ALAP and Minimum are unchanged. The figure retains all six methods. This descriptive view locates notice sensitivity near resource events; the controlled comparison above measures early versus deferred action separately.

C.7 Inner-learning recipe sensitivity

Table 10 holds ASAP, 16-update jobs and the linear task family fixed while crossing Adam/SGD with FIFO/reservoir feedback. All arms retain at most 256 examples, with no additional recent-batch view; SGD has zero momentum and each ordinary update is charged $3P$. The anchor learning rate is .01. A separate development bank selects each recipe's rate from $\{.0003, .001, .003, .01, .03, .1, .3, 1\}$ using two 16,384-interaction scenarios; three new scenarios validate the frozen choices. Confirmation uses ten independent 65,536-interaction scenarios. Identical Adam/FIFO anchor and selected configurations share ten evaluations, giving 70 unique lifetimes and 80 analysis slots.

On this independent confirmation bank, selected SGD/FIFO reaches .78600 versus Adam/FIFO's .78939. Reservoir variants remain lower, with selected rates at the lower search boundary. Adaptation thus depends on the optimizer, replay distribution and learning-rate recipe.



Figure 2: Notice 32 minus notice 64: mean reward differences over entire lifetimes and resource-event windows. Each method is compared with itself across the two notice conditions. Points summarize ten equally weighted scenarios, with five training seeds per learned method; they are descriptive means, without inferential intervals.

Table 10: Inner-recipe confirmation with ASAP: ten scenarios per configuration. The anchor fixes learning rate at .01; selected rates use development data only. All 70 unique lifetimes pass; the Adam/FIFO row is shared between the two comparisons.

Optimizer	Buffer	Anchor reward	Selected LR	Selected reward
Adam	FIFO	0.78939	0.01	0.78939
Adam	Reservoir	0.12043	0.0003	0.04026
SGD	FIFO	0.57669	0.3	0.78600
SGD	Reservoir	0.09610	0.0003	0.24758