COSR: MEMORY-EFFICIENT FP8 TRAINING FOR MULTITASK REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

FP8 reduces compute and memory costs in large-scale deep learning, but bootstrapping, nonstationary targets, and closed-loop feedback between policies and data distributions make deep reinforcement learning sensitive to low-precision perturbations. High-capacity multitask critics further increase training-state memory costs. We find that deterministic writeback in FP8-resident online actor–critic training can suppress updates smaller than the quantization step, degrading long-term performance. Independent stochastic rounding (SR) preserves these updates in expectation but ignores error interactions in forward and backward matrix operations and Adam state updates, allowing locally unbiased noise to accumulate into substantial optimization perturbations. We propose CoSR, an operator-aware joint SR framework. CoSR uses forward/backward channel correlations and the coupled effects of Adam's first and second moments on update directions to coordinate correlations among rounding errors while preserving elementwise rounding probabilities and unbiasedness. It requires no per-parameter compensation buffers, reducing persistent-state memory overhead; major weight matrices and Adam states reside in FP8. Across diverse reinforcement-learning benchmarks, including DeepMind Control Suite and Meta-World, CoSR achieves performance comparable to the high-precision baseline while reducing training-state memory and peak GPU memory by 74% and 63%, respectively. These savings enable wider critics, yielding substantial multitask performance gains under the same hardware budget.

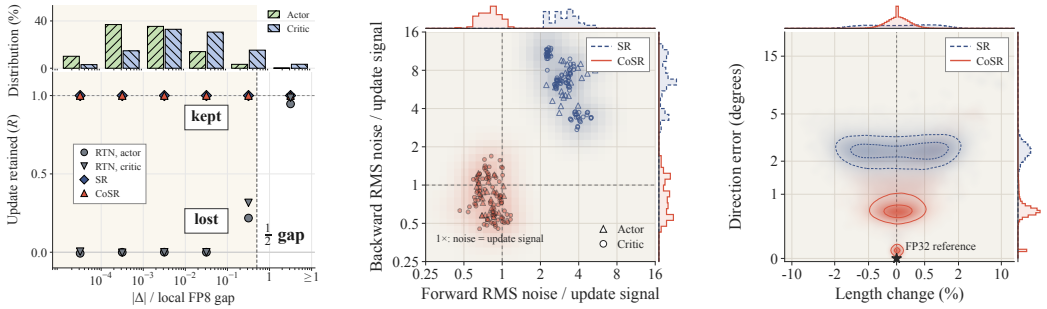
1 INTRODUCTION

Low-precision training reduces computation, data movement, and storage costs in neural network training by reducing the bit widths of numerical representations and arithmetic operations (Micikevicius et al., 2017; Wang et al., 2018). FP8 (Micikevicius et al., 2022) has demonstrated its ability to improve training throughput and reduce GPU memory requirements in large-scale deep learning(Peng et al., 2023; Xi et al., 2025). High-capacity value networks (Nauman et al., 2026; McLean et al., 2025) and mixture-of-experts (MoE) architectures (Hendawy et al., 2024; Kong et al., 2025) have shown benefits in multi-task reinforcement learning. This motivates extending low precision to the main computations and persistent training states of online actor–critic methods to reduce GPU memory and computational costs, thereby enabling larger models to be trained under a fixed hardware budget.

However, this extension entails more than replacing data types. Online actor–critic learning introduces error-propagation paths. The critic must track targets that change with the policy and target network (Lyle et al., 2023; Nikishin et al., 2022), while bootstrapping feeds prediction errors into subsequent temporal-difference targets (Fujimoto et al., 2018; Van Hasselt et al., 2018). Critic errors can also alter action gradients, policy updates, and future data collection, feeding back into value learning (Silver et al., 2014; Lillicrap et al., 2015), affecting learning signals, not only predictions.

Prior work has demonstrated the feasibility of FP16 training in online reinforcement learning (Björck et al., 2021), providing a basis for further reductions in training precision. We use FP8 as the numerical format for storing parameters and the main optimizer states between iterations, reducing reliance on persistent high-precision copies. This further reduces the storage required for training states, creating room to increase critic capacity and support multi-task learning under a fixed hardware budget. This could facilitate on-device adaptation as robots encounter new tasks and environments.

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.



(a) Update retention (b) Weight noise (c) Adam distortion

Figure 1: FP8 writeback. (a) Retained update R vs. $|\Delta|$ in units of the local FP8 gap; RTN drops sub-gap steps, while SR and CoSR keep $R \approx 1$. (b) Forward/backward RMS rounding noise over the update ($1\times$ = noise equals the update). (c) Adam step length and direction vs. an FP32 reference.

Yet FP8-resident training must preserve the small updates that drive learning. Many actor and critic updates are smaller than half the local FP8 gap, as shown in the upper panel of Figure 1a. Deterministic round-to-nearest (RTN) rounds these updates away during writeback (Croci et al., 2022; Zamirai et al., 2020), leaving weights unchanged and preventing small changes from accumulating. This loss appears as zero update retention ($R = 0$) in the lower panel. Stochastic rounding (SR), by contrast, preserves these updates in expectation ($R = 1$) (Croci et al., 2022; Gupta et al., 2015), allowing even these small changes to contribute to learning.

But preserving scalar expectations does not control how rounding errors combine (Suresh et al., 2022). In actor and critic networks, weight errors combine across channels, perturbing forward outputs and backpropagated input gradients. The blue SR points in Figure 1b exceed $1\times$ on both axes: rounding noise exceeds the update-induced signal. For Adam (Kingma & Ba, 2014), first- and second-moment errors interact through nonlinear normalization, distorting update length and direction. As figure 1c shows, SR's blue distribution lies away from the FP32 reference at the origin.

To address these computational errors while retaining expected updates, we propose CoSR, an operator-aware framework for joint stochastic rounding. CoSR preserves each element's SR distribution while organizing correlations among rounding errors to promote cancellation in the relevant operations. ***For the actor and critic***, it coordinates weight rounding using forward and backward channel correlations. ***For Adam***, it jointly rounds the two moment estimates according to their coupled effect on the normalized update. The three panels illustrate the goal of this design: preserving expected update retention (Figure 1a) while reducing forward/backward noise and bringing Adam updates closer to the FP32 reference relative to independent SR (Figures 1b and 1c). For the slowly tracking target network, we additionally use relative lag encoding to improve the resolution of soft updates. Together, these components keep the main training states resident in FP8 using compact statistics and no dense compensation buffers, reducing memory and computation cost and enabling larger models under fixed hardware budget. Contributions are as follows:

- **Operator-aware joint rounding.** We propose CoSR, which preserves SR marginals and provably minimizes forward/backward perturbation risks within fixed 2×2 blocks and coordinatewise linearized Adam-update error. With relative target encoding, it keeps main training states in FP8 without dense compensation.
- **Effective and efficient FP8 training.** CoSR performs comparably to BRC-FP32 on Meta-World-50 (Yu et al., 2020b), DMC-Hard (Tassa et al., 2018), and HumanoidBench-Medium (Sferrazza et al., 2024). Using Blackwell-native FP8 tensor operations on NVIDIA RTX PRO 6000 GPUs, CoSR reduces persistent-state and peak GPU memory by 73.69% and 62.84%, respectively, while lowering update latency and maintaining high-precision-level performance.
- **Scaling under a fixed memory budget.** At comparable peak training memory, CoSR doubles critic width (4096 to 8192), improving MT50, DMC, and ShadowHand scores by 11%–14% in 165-task joint training without extra interactions. This demonstrates the practical value of low-precision RL by translating memory savings into greater model capacity under real hardware constraints.

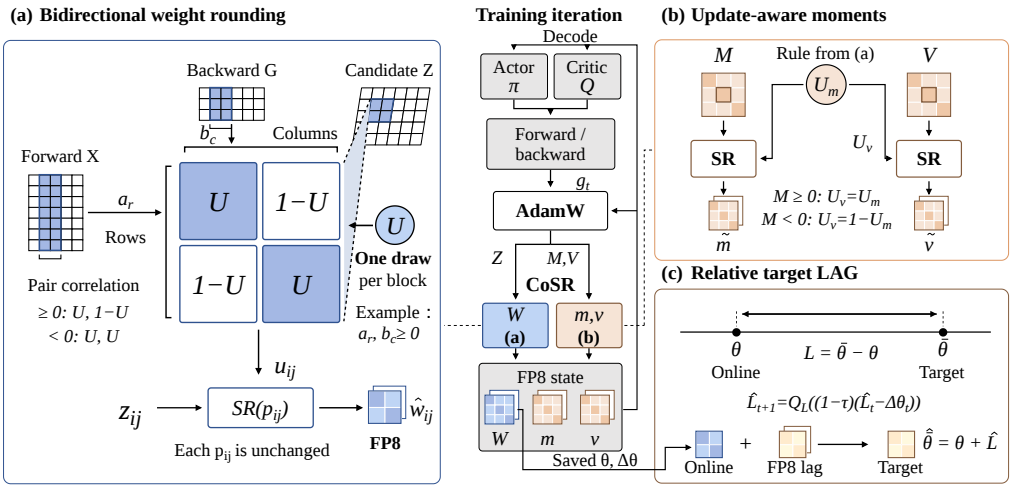


Figure 2: Overview of CoSR. (a) Weight rounding couples randomness using forward and backward correlations; (b) moment rounding couples AdamW moments according to the first-moment sign; (c) relative target LAG stores the target–online parameter difference in FP8 for target reconstruction.

2 Related Work

2.1 Multi-Task Reinforcement Learning

Multi-task reinforcement learning aims to solve multiple tasks with a shared agent (Yang et al., 2020; Teh et al., 2017; Sodhani et al., 2021). A central challenge is to transfer useful knowledge while limiting interference between tasks. PCGrad Yu et al. (2020a) and PaCo Sun et al. (2022) address this through gradient projection and learned parameter sharing, respectively. Among MoE approaches, MOORE promotes diverse shared representations (Hendawy et al., 2024), while M3DT scales offline multi-task learning through expert expansion (Kong et al., 2025). BRC Nauman et al. (2026) further shows that high-capacity, regularized value networks can support effective online learning across diverse tasks. Building on BRC, we study how FP8 training supports larger multitask models and stronger performance under limited memory.

2.2 Low-Precision Training and Rounding

Low-precision training reduces computational cost Micikevicius et al. (2017); Wang et al. (2018) and saves memory by compressing optimizer states, gradients, and activations Dettmers et al. (2021); Peng et al. (2023); Xi et al. (2025). However, quantization can discard small updates. ECO Nikdan et al. (2026) compensates weight quantization errors through momentum feedback, FlashOptim Ortiz et al. (2026) encodes weight residuals with integer correction codes and reconstructs weights during updates, while SR Croci et al. (2022) preserves small updates in expectation. Rounding can target computational goals: Correlated Quantization Suresh et al. (2022) coordinates randomness to reduce distributed mean estimation error, whereas signed-SR_ε Xia et al. (2022) biases rounding toward descent. CoSR coordinates unbiased rounding errors based on downstream computations.

2.3 Low-Precision Reinforcement Learning

FP16RL Björck et al. (2021) stabilizes half-precision SAC through numerical reformulations and Kahan compensation for critic parameter updates and target soft updates. ActorQ in QuaRL Krishnan et al. (2019) quantizes rollout policies and parameter broadcasts while retaining a full-precision learner. Jet-RL Xi et al. (2026) aligns FP8 quantization between LLM training and rollouts while retaining BF16 master weights. We study repeated FP8 writeback of persistent states in online actor–critic training. Alongside coordinated weight and Adam-state rounding, LAG encodes targets relative to online parameters to better represent their slow soft updates.

3 METHOD

To address small-update information loss under deterministic rounding and joint errors overlooked by independent SR, we propose **CoSR (Coordinated Stochastic Rounding)**, an operator-aware joint FP8 rounding framework (Figure 2) for online actor–critic training. CoSR preserves elementwise SR probabilities while coordinating rounding errors according to subsequent computations. With the main weight matrices and Adam states stored as one FP8 code per element and no dense compensation, CoSR matches high-precision training performance at low storage and computational cost.

3.1 PRELIMINARIES

For the actor π_ϕ and online critic Q_θ, we store the main weight matrices and their Adam moments in FP8:

$$W = s_W \odot C_W, \qquad C_W \in \mathbb{F}_8, \tag{1}$$

$$\widetilde{m} = s_m \odot C_m, \qquad \widetilde{v} = s_v \odot C_v. \tag{2}$$

$\mathbb{F}_8$ denotes the FP8 value set, each state uses a separate scale, and $\odot$ broadcasts scales over blocks.

We retain the actor–critic objectives (Haarnoja et al., 2018) and AdamW updates (Loshchilov & Hutter, 2017), use FP8 GEMM operands (Sun et al., 2019), and apply CoSR at writeback. Rounding uses decoded values on scale-defined grids (Rouhani et al., 2023). For z between adjacent grid points $z^- < z^+$, SR returns z^+ if $U < p(z) = (z - z^-)/(z^+ - z^-)$, and z^- otherwise, where $U \sim \mathcal{U}(0, 1)$; exactly representable values are unchanged. Our analysis conditions on grids selected before sampling and assumes no saturation.

For fixed candidates, quantization grids, and coefficients a_i, rounding errors $e_i = \widehat{z}_i - z_i$ induce a dot-product perturbation $\delta = \sum_i a_i e_i$, with

$$\mathbb{E}[\delta^2] = \sum_i a_i^2 \mathbb{E}[e_i^2] + 2 \sum_{i<j} a_i a_j \mathbb{E}[e_i e_j]. \tag{3}$$

With fixed SR marginals, the first term is constant, so negative weighted cross terms reduce the mean-squared perturbation. CoSR couples sampling using forward and backward channel correlations and local sensitivities of the update direction to Adam moments, promoting error cancellation while preserving elementwise unbiasedness.

3.2 BIDIRECTIONAL WEIGHT ROUNDING

3.2.1 FORWARD AND BACKWARD RISKS

Consider a linear layer $Y = XW$, where $X \in \mathbb{R}^{B \times d_{\text{in}}}$ is the input, $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ is the weight matrix, and B is the batch size. Denote the output cotangent by $G = \partial \mathcal{L}/\partial Y \in \mathbb{R}^{B \times d_{\text{out}}}$. Let Z be the candidate weights produced by the optimizer and $\widehat{W}$ be the weights after writeback, with rounding error $E = \widehat{W} - Z$.

Fixing the effective forward input X and backward operand G recorded in the current step, the weight error causes the following respective perturbations, from which we define the forward and backward-input perturbation risks:

$$\begin{aligned} \Delta Y = XE, \quad & \mathcal{R}_f = \mathbb{E}\|XE\|_F^2, \\ \Delta(dX) = GE^\top, \quad & \mathcal{R}_b = \mathbb{E}\|GE^\top\|_F^2. \end{aligned} \tag{4}$$

The expectations here are taken only over the rounding randomness newly generated in the current step. The actor and critic use the same risk definitions, but each uses X, G produced by its own network and loss.

Expanding these two objectives gives

$$\begin{aligned} \|XE\|_F^2 &= \operatorname{tr}(E^\top X^\top X E), \\ \|GE^\top\|_F^2 &= \operatorname{tr}(E G^\top G E^\top). \end{aligned} \tag{5}$$

Thus, the input-channel inner products $X^\top X$ and output-gradient-channel inner products $G^\top G$ naturally become the coefficients of the rounding-error cross terms. We do not explicitly construct the full Gram matrices, but extract only the inner products of adjacent channels:

$$a_r = \langle X_{:,2r}, X_{:,2r+1} \rangle , \qquad b_c = \langle G_{:,2c}, G_{:,2c+1} \rangle . \tag{6}$$

a_r controls the rounding relationship between weights in two adjacent rows of the same column to reduce forward perturbations; b_c controls the rounding relationship between weights in two adjacent columns of the same row to reduce backward-input perturbations. The statistics are computed directly from the current forward and backward operands and reused for the current parameter writeback.

3.2.2 Local Random Coupling

A 2×2 block is the smallest rectangular block containing both a same-column pair relevant to the forward risk and a same-row pair relevant to the backward-input risk. The signs of the associated channel inner products determine the preferred dependence, while their magnitudes weight the corresponding pairwise contributions to the risks.

We partition the weight matrix into non-overlapping 2×2 blocks. For each block (r, c), we independently generate $U_{rc} \sim \mathcal{U}(0, 1)$ and define row and column flip flags $f_r = \mathbf{1}[a_r \geq 0]$ and $k_c = \mathbf{1}[b_c \geq 0]$. For positions $i, j \in \{0, 1\}$ within a block, let $\phi_{ij} = (if_r) \oplus (jk_c)$, and construct

$$V_{2r+i,\,2c+j} = \begin{cases} U_{rc}, & \phi_{ij} = 0, \\ 1 - U_{rc}, & \phi_{ij} = 1. \end{cases} \tag{7}$$

where $\oplus$ denotes exclusive OR. Each position then performs rounding using its own upward-rounding probability p_{ij}.

When the input inner product is positive, adjacent weights in the same column use complementary random numbers; when it is negative, they use a common random number. The output-gradient inner product determines the random-number relationship between weights in the same row according to the same rule. For example, when both a_r, b_c are positive, the random-number layout is $\left[\begin{smallmatrix} U_{rc} & 1-U_{rc} \\ 1-U_{rc} & U_{rc} \end{smallmatrix}\right]$.

This layout simultaneously satisfies the vertical and horizontal complementarity requirements. Since both U_{rc} and $1 - U_{rc}$ follow a uniform distribution, each weight retains its original SR marginal distribution. Rounding within a block is jointly determined by the forward and backward statistics, but the weights are rounded only once, during persistent writeback after the optimizer update.

3.2.3 Bidirectional Perturbation Guarantees

Proposition 1: Bidirectionally Optimal Coupling within a Fixed Local Block Partition. Fix X, G, Z and the quantization grid. Consider all joint rounding distributions that preserve the scalar SR marginals, use the same 2×2 block partition, and are independent across blocks; denote this set by $\mathcal{P}_2$. The CoSR construction $P^\star$ in equation 7 simultaneously satisfies

$$\begin{aligned} \mathcal{R}_f(P^\star) &= \min_{P \in \mathcal{P}_2} \mathcal{R}_f(P), \\ \mathcal{R}_b(P^\star) &= \min_{P \in \mathcal{P}_2} \mathcal{R}_b(P). \end{aligned} \tag{8}$$

In particular, independent SR belongs to $\mathcal{P}_2$, and therefore $\mathcal{R}_f^{\mathrm{CoSR}} \leq \mathcal{R}_f^{\mathrm{SR}}$ and $\mathcal{R}_b^{\mathrm{CoSR}} \leq \mathcal{R}_b^{\mathrm{SR}}$.

The two vertical edges of a block share a_r, and the two horizontal edges share b_c. Therefore, the row and column flips in equation 7 can simultaneously attain the covariance extrema required by all four edges. Covariances between diagonal positions do not enter either risk and thus introduce no additional conflicts. The full proof is provided in Appendix C.2.

With the scalar SR marginals fixed, CoSR and independent SR have the same expected weight reconstruction error $\mathbb{E}\|E\|_F^2$. Their operator risks can nevertheless differ because the entrywise errors interact differently through the cross-channel terms in $X^\top X$ and $G^\top G$. These guarantees therefore reflect **reorganizing error directions at the same scalar rounding precision**.

3.3 UPDATE-AWARE ADAM STATE ROUNDING

Network weights affect learning through matrix multiplication, whereas the first- and second-moment Adam states jointly determine the preconditioned update. Separately minimizing the reconstruction errors of m, v does not necessarily minimize the update error produced by their combination. We therefore apply CoSR to the Adam states of the actor and critic.

3.3.1 ADAMW UPDATE SEMANTICS

Let the current decoded states be $\widetilde{m}_t, \widetilde{v}_t$ and the gradient be g_t. Denote the bias-correction factors by $c_{1,t+1} = 1 - \beta_1^{t+1}$ and $c_{2,t+1} = 1 - \beta_2^{t+1}$. The candidate states, update direction, and weight candidate are computed according to the original AdamW rule(Loshchilov & Hutter, 2017):

$$\begin{aligned} M_{t+1} &= \beta_1 \widetilde{m}_t + (1-\beta_1) g_t, \\ V_{t+1} &= \beta_2 \widetilde{v}_t + (1-\beta_2) g_t^2, \\ u_{t+1} &= \frac{M_{t+1}/c_{1,t+1}}{\sqrt{V_{t+1}/c_{2,t+1}} + \epsilon}, \\ Z_{t+1} &= W_t - \eta_t \left(u_{t+1} + \lambda W_t \right). \end{aligned} \tag{9}$$

where η_t is the learning rate and λ is the weight decay coefficient. This process preserves Adam's moment estimation and bias-correction structure.

We first compute the current weight candidate Z_{t+1} using M_{t+1}, V_{t+1} before they are requantized, and then jointly round the moment states and store them for subsequent iterations. Thus, CoSR changes how the recursive states are written back, rather than adding a new correction term to the original AdamW update.

3.3.2 JOINT MOMENT ROUNDING

For a single coordinate, let $F(M, V)$ denote the Adam direction map in equation 9. To guide moment writeback for subsequent iterations, we use its linearized distortion under writeback errors e_m, e_v as a local surrogate:

$$\delta u_{\text{lin}} = \alpha e_m + \beta e_v, \qquad \alpha = \frac{\partial F}{\partial M}, \quad \beta = \frac{\partial F}{\partial V}. \tag{10}$$

When $V > 0$, $\alpha > 0$ and $\operatorname{sign}(\beta) = -\operatorname{sign}(M)$. Therefore, when $M > 0$, errors of the same sign in the first and second moments have opposite effects on the update direction; when $M < 0$, the corresponding beneficial relationship becomes oppositely directed coupling. We use

$$U_v = \begin{cases} U_m, & M \geq 0, \\ 1 - U_m, & M < 0. \end{cases} \tag{11}$$

We round M, V to FP8 using separate scales and their respective SR probabilities. We construct the spatial field U_m using Section 3.2 and obtain U_v from equation 11.

Proposition 2: Optimal Coupling for the Linearized Adam Surrogate. Fix candidate moments M, V with $V > 0$, their quantization grids, and scalar SR marginals. Among all couplings of e_m, e_v at a single coordinate, equation 11 minimizes $\mathbb{E}[(\delta u_{\text{lin}})^2]$, and therefore

$$\mathbb{E}\left[(\delta u_{\text{CoSR,lin}})^2\right] \leq \mathbb{E}\left[(\delta u_{\text{SR,lin}})^2\right]. \tag{12}$$

This follows from the same covariance-extremum principle as Proposition 1; see Appendix C.3 for the proof.

The choice between shared and complementary uniforms at each coordinate depends only on the sign of M. Sensitivity magnitudes affect the surrogate risk, but its minimizing covariance is selected by the sign of $\alpha\beta$. Each moment retains its own scale and SR probability, so the construction coordinates writeback through the update map without altering either scalar distribution.

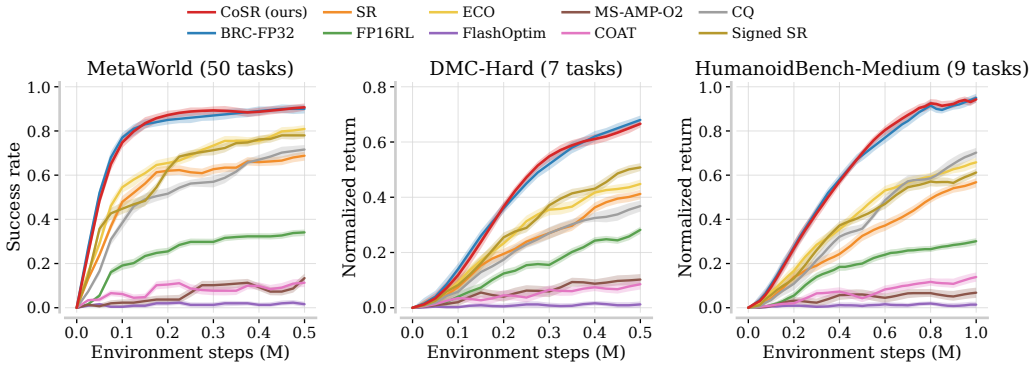


Figure 3: Learning curves on the three benchmarks. Curves and shaded regions represent the means over five random seeds and the 95% confidence intervals, respectively.

3.4 Storage Overhead

For a weight matrix, the rule for each input-channel pair is shared by all output columns, and the rule for each output-gradient-channel pair is shared by all input rows. Therefore, a $d_{\text{in}} \times d_{\text{out}}$ weight matrix requires only $d_{\text{in}}/2 + d_{\text{out}}/2$ sign statistics, without storing a compensation matrix of the same shape. Rounding random numbers are generated based on the training step and parameter identifier and are not retained as historical states. The same applies to coordinated rounding of Adam states. The auxiliary statistics encode relationships between rounding decisions rather than numerical corrections to individual parameters. For a square weight matrix of width d, the construction uses d sign statistics to coordinate d^2 entries. Their storage therefore grows linearly with layer width, while the weight and moment matrices grow quadratically, making the relative metadata footprint smaller for wider layers.

3.5 Lag-Based Target Soft Updates

The target critic constructs learning targets using only forward propagation and follows the online critic through $\bar{\theta}_{t+1} = (1-\tau)\bar{\theta}_t + \tau\theta_{t+1}$. Direct FP8 storage must resolve small increments $\tau(\theta_{t+1} - \bar{\theta}_t)$ at the full parameter scale. FP16 RL (Björck et al., 2021) uses Kahan summation (Kahan, 1965) with a compensation matrix as large as the network. We instead store the target's relative lag with one FP8 code per element and a separate dynamic scale:

$$L_t = \bar{\theta}_t - \theta_t, \qquad \widehat{\bar{\theta}}_t = \theta_t + \widehat{L}_t. \tag{13}$$

The networks share their initial parameters, giving $L_0 = 0$. We reuse existing online parameters as the reconstruction reference and define $\Delta\theta_t = \theta_{t+1} - \theta_t$ from **the actual online displacement after CoSR writeback**. This keeps the lag and its reference consistent. The lag update and reconstructed target satisfy

$$\widehat{L}_{t+1} = Q_L\Big((1-\tau)(\widehat{L}_t - \Delta\theta_t)\Big), \tag{14}$$

$$\widehat{\bar{\theta}}_{t+1} = (1-\tau)\widehat{\bar{\theta}}_t + \tau\theta_{t+1} + \varepsilon^L_{t+1}. \tag{15}$$

Here, Q_L encodes the lag in FP8, and ε^L_{t+1} is its quantization error. Before quantization, the recurrence is algebraically equivalent to the original soft update (Appendix C.4).

When the target closely tracks the online critic, lag encoding uses a dynamic range matched to their difference rather than to the full parameter values. Subtracting $\Delta\theta_t$ re-expresses the previous reconstructed target relative to the updated online weights without changing that target; multiplication by $1-\tau$ then performs the soft update in this representation. With fixed online parameters, the unquantized lag contracts by $1-\tau$, expressing the target's approach to the online reference. CoSR coordinates writeback errors, while LAG adapts the encoding reference to target soft updates.

4 EXPERIMENTS

We implement CoSR within BRC Nauman et al. (2026), a multitask reinforcement learning framework with high-capacity critics, and evaluate it across multiple benchmarks against representative baselines.

4.1 EXPERIMENTAL SETUP

Benchmarks and Protocol. Following BRC's task splits and training settings, we use MetaWorld-50 (MT50), DMC-Hard, and HumanoidBench-Medium (HB-Medium), with per-task budgets of 500K, 500K, and 1M steps, respectively. All experiments use five seeds, with matched architectures, batch sizes, and update frequencies within each benchmark. Evaluation uses 10 episodes per task every 25K steps. All experiments run on NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs.

Comparison Methods. Except for BRC-FP32, all baselines use FP8 main-weight storage within BRC. We compare with high-precision BRC (BRC-FP32) Nauman et al. (2026), SR, the half-precision RL method FP16RL Björck et al. (2021), the training state compression methods ECO Nikdan et al. (2026), FlashOptim Ortiz et al. (2026), MS-AMP O2 (FP8-LM) Peng et al. (2023), and COAT Xi et al. (2025), and the rounding methods Correlated Quantization Suresh et al. (2022) and signed-SR$_\varepsilon$ Xia et al. (2022). For fair comparison, Appendix Table 6 reports the FP8-adapted baseline configurations, while Table 12 provides additional results under the methods' original precision settings.

Evaluation Metrics. We report final success rates on MT50 and normalized returns on DMC-Hard and HB-Medium. Scores are averaged equally across tasks within each seed, then across seeds, with 95% confidence intervals obtained by bootstrapping seed-level averages Agarwal et al. (2021). We measure persistent state size, peak GPU memory under matched configurations, and synchronized update latency after warmup. State size includes weights, Adam moments, targets, scales, and master or compensation states; latency excludes compilation, environment interaction, and evaluation.

4.2 PERFORMANCE AND RESOURCE

Figure 4 compares persistent-state memory, update latency, and mean performance, computed by equally averaging the final-score ratios to BRC-FP32 across the three benchmarks. Bubble area represents persistent-state memory, and the horizontal axis reports MT50 update latency. Several low-precision methods have similar state sizes but markedly different learning performance. CoSR remains close to the high-precision performance reference while requiring smaller persistent states and lower update latency than BRC-FP32, achieving an excellent balance between learning performance and required resources overall.

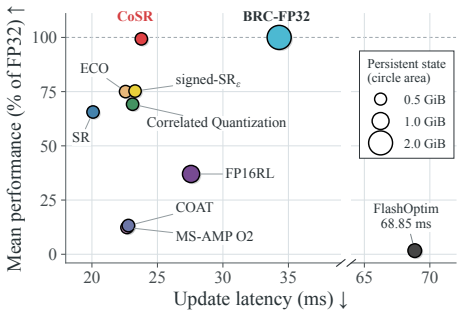


Figure 4: Performance–resource trade-off across methods.

On MT50, CoSR reduces persistent-state memory and peak GPU memory by 73.69% and 62.84%. Peak usage includes temporary buffers, whereas persistent-state size measures the storage occupied by the retained training states. This enables us to train wider models within the same hardware budget4.4.

Figure 3 shows CoSR tracking BRC-FP32 and outperforming other low-precision baselines in our FP8 storage setting. Several mechanisms may contribute. Independent SR is scalar-unbiased but leaves weighted errors uncoordinated. FP16RL's FP8 compensation may not fully recover small updates. ECO can lose error feedback when requantizing corrected momentum. FlashOptim's grid coarsens with FP8 weight spacing, so small updates may round away. Quantizing MS-AMP O2's second moments from FP16 to FP8 may perturb Adam's denominator. COAT's separate moment expansions can compound residual errors in $m/(\sqrt{v}+\epsilon)$. Correlated Quantization promotes cancellation in averages, not necessarily under signed, nonuniform network coefficients. signed-SR$_\varepsilon$ can reinforce inaccurate descent directions when critic quantization distorts gradients. CoSR preserves SR marginals, coordinating weight and moment errors for downstream computations; LAG better represents slow target updates. Detailed result table is in AppendixD.1.

4.3 Ablation Study

Table 1 evaluates all 2^3 combinations of coordinated weight rounding, coordinated Adam-state rounding, and LAG on MetaWorld-50 (MT50). The weight and Adam-state settings apply to both the actor and critic. Checkmarks indicate enabled modules. Disabled rounding modules use independent SR, while disabling LAG stores absolute target parameters in FP8. We report final success rates averaged equally over 50 tasks and five seeds. All configurations share the FP8 target format, scale update rule, target rounding strategy, and forward computation precision.

Weights	Adam states	LAG	Success (%) ↑
			$68.8_{[66.5,70.6]}$
✓			$72.5_{[71.0,73.6]}$
	✓		$75.6_{[72.5,76.7]}$
		✓	$75.4_{[73.9,77.3]}$
✓	✓		$83.7_{[81.2,86.1]}$
✓		✓	$84.2_{[82.5,86.9]}$
	✓	✓	$84.8_{[82.1,87.0]}$
✓	✓	✓	$\mathbf{90.7}_{[89.2,92.5]}$

Table 1: MT50 ablation. Five-seed means; subscripts: 95% CIs.

Single-module configurations achieve 72.5–75.6% success and two-module combinations reach 83.7–84.8%, versus 90.7% for the full configuration. Weight rounding affects predictions and gradients, Adam-state rounding affects preconditioned updates, and LAG affects target tracking and hence bootstrapped targets. Disabling each module in turn lowers success to 84.8%, 84.2%, and 83.7%: drops of 5.9, 6.5, and 7.0 percentage points, respectively. LAG raises success from 68.8% to 75.4% alone and from 83.7% to 90.7% with both rounding modules enabled. Thus, coordination improves learning beyond independent SR, with further gains from LAG.

4.4 Scaling under Memory Budget

To evaluate model scaling under a fixed memory budget, we compare BRC-FP32 with a critic width of 4096 against CoSR with a width of 8192, using the former's peak JAX-allocated training memory as the budget reference. Both models jointly train on 165 tasks: 50 from MT50, 30 from DMC, and 85 from ShadowHand(Huang et al., 2021). All other experimental settings are identical, with 1M environment steps per task. To control for model width, we additionally evaluate BRC-FP32 and independent SR at a critic width of 8192, with results reported in Appendix D.4.

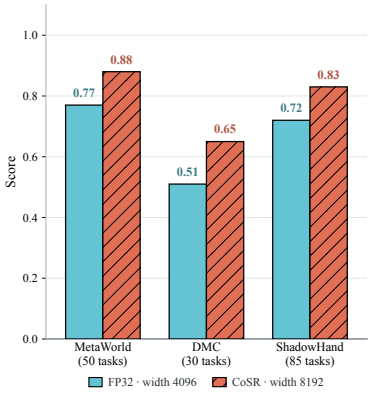


Figure 5: Scaling on 165 tasks at same peak GPU memory.

Figure 5 shows improvements across all three task groups at approximately matched peak memory. MT50 success increases from 0.77 to 0.88, while DMC and ShadowHand average scores rise from 0.51 to 0.65 and from 0.72 to 0.83. CoSR thus doubles the critic width and improves multi-task performance without additional environment interactions. Since peak memory includes activations and temporary buffers of the wider network, these results demonstrate that persistent-state savings translate into greater model capacity within the same memory budget.

5 Conclusion

We presented CoSR, an operator-aware joint rounding framework for FP8 online actor–critic training. CoSR preserves elementwise SR probabilities while coordinating weight and Adam-state rounding errors according to downstream computations. LAG complements this design by encoding target parameters relative to online parameters, improving the representation of slow soft updates. Experiments on MetaWorld-50, DMC-Hard, and HumanoidBench-Medium demonstrate near-FP32 performance with reduced persistent-state memory, peak memory, and update latency. CoSR enables larger, more capable multitask models under the same memory budget through low-precision training.

Limitation. Activation and temporary-buffer costs still constrain model scaling despite persistent-state savings. Our evaluation focuses on FP8 online actor–critic training; extending CoSR to other training paradigms and precision formats remains future work.

AI USE STATEMENT

Generative AI tools were used to standardize the formatting of tables in this paper.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Johan Björck, Xiangyu Chen, Christopher De Sa, Carla P Gomes, and Kilian Weinberger. Low-precision reinforcement learning: running soft actor-critic in half precision. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2021.

Samarth Brahmbhatt, Cusuh Ham, Charles C Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8701–8711. IEEE, 2019.

Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, 2015.

Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3): 211631, 2022.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. Pmlr, 2018.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *International Conference on Learning Representations*, volume 2024, pp. 6344–6365, 2024.

Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.

William Kahan. Pracniques: further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yilun Kong, Guozheng Ma, Qi Zhao, Haoyu Wang, Li Shen, Xueqian Wang, and Dacheng Tao. Mastering massive multi-task reinforcement learning via mixture-of-expert decision transformer. *arXiv preprint arXiv:2505.24378*, 2025.

Srivatsan Krishnan, Maximilian Lam, Sharad Chitlangia, Zishen Wan, Gabriel Barth-Maron, Aleksandra Faust, and Vijay Janapa Reddi. Quarl: Quantization for fast and environmentally sustainable reinforcement learning. *arXiv preprint arXiv:1910.01055*, 2019.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International conference on machine learning*, pp. 23190–23211. PMLR, 2023.

Reginald McLean, Evangelos Chatzaroulas, Jordan Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Multi-task reinforcement learning enables parameter scaling. *arXiv preprint arXiv:2503.05126*, 2025.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *Advances in Neural Information Processing Systems*, 38:50818–50858, 2026.

Mahdi Nikdan, Amir Zandieh, Dan Alistarh, and Vahab Mirrokni. Eco: Quantized training without full-precision master weights. *arXiv preprint arXiv:2601.22101*, 2026.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Jose Javier Gonzalez Ortiz, Abhay Gupta, Christopher Rinard, and Davis Blalock. Flashoptim: Optimizers for memory-efficient training. *arXiv preprint arXiv:2602.23349*, 2026.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International conference on machine learning*, pp. 9767–9779. PMLR, 2021.

Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 21495–21507, 2022.

Xiao Sun, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Ananda Theertha Suresh, Ziteng Sun, Jae Ro, and Felix Yu. Correlated quantization for distributed mean estimation and optimization. In *International Conference on Machine Learning*, pp. 20856–20876. PMLR, 2022.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. Coat: Compressing optimizer states and activations for memory-efficient fp8 training. In *International Conference on Learning Representations*, volume 2025, pp. 42989–43011, 2025.

Haocheng Xi, Charlie Ruan, Peiyuan Liao, Yujun Lin, Han Cai, Yilong Zhao, Shuo Yang, Kurt Keutzer, Song Han, and Ligeng Zhu. Jet-rl: Enabling on-policy fp8 reinforcement learning with unified training and rollout precision flow. *arXiv preprint arXiv:2601.14243*, 2026.

Lu Xia, Stefano Massei, Michiel E Hochstenbach, and Barry Koren. On the influence of stochastic roundoff errors and their bias on the convergence of the gradient descent method with low-precision floating-point computation. *arXiv preprint arXiv:2202.12276*, 2022.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020a.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020b.

Pedram Zamirai, Jian Zhang, Christopher R Aberger, and Christopher De Sa. Revisiting bfloat16 training. *arXiv preprint arXiv:2010.06192*, 2020.

A CoSR Training Algorithm

Algorithm 1 summarizes one learner update with coordinated FP8 writeback for the weights and Adam states of both the actor and critic. Each parameter update uses unrounded moment candidates, while LAG uses the critic's actual displacement after weight writeback.

Algorithm 1 One CoSR learner update

Require: Minibatch $\mathcal{B}$; actor ϕ, critic θ, lag $\widehat{L}$; Adam states $(m_\phi, v_\phi, m_\theta, v_\theta)$; temperature α.

Compute the critic loss using target parameters $\bar{\theta} = \theta + \widehat{L}$.
Backpropagate to obtain g_θ and per-layer operands (X_θ, G_θ).
$(Z_\theta, M_\theta, V_\theta) \leftarrow \mathrm{AdamW}(\theta, m_\theta, v_\theta, g_\theta)$.
$\theta^+ \leftarrow \mathrm{SR}_8\big(Z_\theta, \mathrm{Couple}(X_\theta, G_\theta)\big)$.
$U_m \leftarrow \mathrm{Couple}(X_\theta, G_\theta)$; $U_v \leftarrow \mathrm{where}(M_\theta < 0, 1 - U_m, U_m)$.
$(m_\theta, v_\theta) \leftarrow \big(\mathrm{SR}_8(M_\theta, U_m), \mathrm{SR}_8(V_\theta, U_v)\big)$.
$\widehat{L} \leftarrow \mathrm{RTN}_8\big((1-\tau)[\widehat{L} - (\theta^+ - \theta)]\big)$; $\theta \leftarrow \theta^+$.
Compute the actor loss using θ and obtain g_ϕ and (X_ϕ, G_ϕ).
$(Z_\phi, M_\phi, V_\phi) \leftarrow \mathrm{AdamW}(\phi, m_\phi, v_\phi, g_\phi)$.
$\phi \leftarrow \mathrm{SR}_8\big(Z_\phi, \mathrm{Couple}(X_\phi, G_\phi)\big)$.
$U_m \leftarrow \mathrm{Couple}(X_\phi, G_\phi)$; $U_v \leftarrow \mathrm{where}(M_\phi < 0, 1 - U_m, U_m)$.
$(m_\phi, v_\phi) \leftarrow \big(\mathrm{SR}_8(M_\phi, U_m), \mathrm{SR}_8(V_\phi, U_v)\big)$.
Update α using the policy entropy.
function $\mathrm{Couple}(X, G)$
    $a_r \leftarrow \langle X_{:,2r}, X_{:,2r+1} \rangle$; $b_c \leftarrow \langle G_{:,2c}, G_{:,2c+1} \rangle$.
    Draw fresh independent $u_{rc} \sim \mathcal{U}(0,1)$ for each 2×2 block.
    $f^{rc}_{ij} \leftarrow \big(i\mathbf{1}[a_r \geq 0]\big) \oplus \big(j\mathbf{1}[b_c \geq 0]\big)$, $i, j \in \{0, 1\}$.
    return U, where $U_{2r+i,2c+j} = (1 - f^{rc}_{ij})u_{rc} + f^{rc}_{ij}(1 - u_{rc})$.
end function

Operations apply per selected FP8 matrix; other parameters retain their configured updates and precisions. Arithmetic uses decoded values. AdamW returns candidate weights and unrounded moments. $\mathrm{SR}_8(Z, U)$ uses U as the uniform thresholds for elementwise stochastic rounding, while RTN_8 rounds to nearest. Both write FP8 codes with refreshed FP32 scales: per matrix for weights and lag, and per 128-element block for each moment.

B Benchmarks, baselines, and experimental settings

Our main comparison uses three multi-task settings: Meta-World-50 (MT50), DMC-Hard, and HumanoidBench-Medium (HB-Medium). The capacity-scaling experiment jointly trains on 165 tasks from MT50, a 30-task DMC collection, and an 85-object ShadowHand collection. Table 2 summarizes the training budgets. The task collections follow BRC (Nauman et al., 2026); the original benchmarks and their roles in our evaluation are described below.

Table 2: Training budgets per random seed. Total interactions are the number of tasks multiplied by the per-task training budget and exclude evaluation episodes. The two DMC collections serve different experiments.

Setting	Tasks	Steps/task	Total steps	Reported metric
MT50	50	500K	25M	Success rate
DMC-Hard	7	500K	3.5M	Normalized return
HB-Medium	9	1M	9M	Normalized return
Joint capacity scaling	165	1M	165M	Separate score for each task group

B.1 Meta-World-50

Meta-World is a MuJoCo-based collection of 50 robotic manipulation tasks with a shared tabletop setting and robot embodiment (Yu et al., 2020b; Todorov et al., 2012). Its tasks include reaching and object transport, opening and closing articulated objects, pressing buttons, inserting pegs, and using tools. Although many tasks share motions such as reaching and grasping, their contact patterns and

goal conditions differ. This combination makes MT50 useful for evaluating whether a shared agent can retain diverse skills while reusing common representations.

We use the multi-task MT50 setting, in which all 50 tasks participate in both training and evaluation. The agent is evaluated on these trained tasks, rather than on the held-out tasks of a meta-learning split. Each task receives 500K training interactions in the main comparison and 1M in the joint capacity-scaling experiment. We report the equally weighted mean of the task success rates. Figure 6 illustrates the task diversity; Table 3 lists the task names.

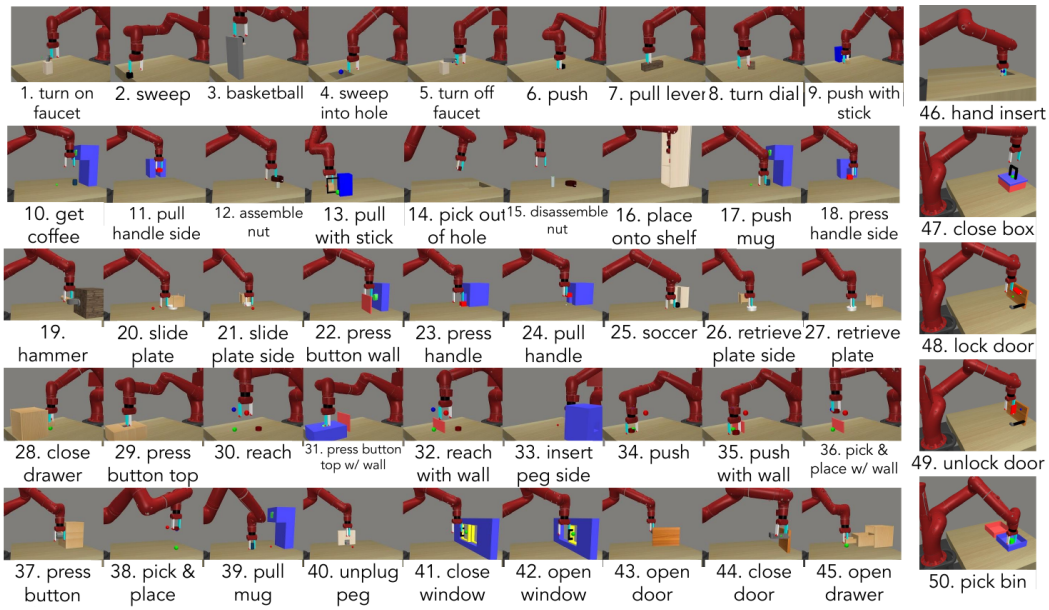


Figure 6: The 50 Meta-World tasks. Task panels are reproduced from Figure 1 of Yu et al. (2020b), with the original split headings omitted. All 50 tasks shown here belong to our MT50 training and evaluation set; the original figure's meta-learning split does not apply to this setting.

B.2 DeepMind Control Suite

The DeepMind Control Suite provides continuous-control environments in MuJoCo, spanning different articulated bodies and control objectives (Tassa et al., 2018; Tunyasuvunakool et al., 2020). Tasks within a domain reuse a body while varying the required behavior, such as maintaining a standing posture or moving at a target speed. This permits comparisons across both related behaviors and different dynamics.

Our main comparison uses DMC-Hard: four dog tasks (`stand`, `walk`, `trot`, and `run`) and three humanoid tasks (`stand`, `walk`, and `run`). These seven tasks require coordinated control of the dog or humanoid body. We allocate 500K interactions per task and report the mean normalized return.

The joint 165-task experiment instead uses the 30 DMC tasks listed in Table 4. This collection includes the task extensions used in BRC's diverse-embodiment setting, including backward locomotion and three-link reaching. These 30 tasks receive 1M interactions each. Their task list is distinct from the seven-task DMC-Hard collection.

B.3 HumanoidBench-Medium

HumanoidBench provides simulated whole-body control tasks involving locomotion and manipulation (Sferrazza et al., 2024). The full benchmark has 27 tasks and supports different robot and end-effector configurations. Its environments combine balance, coordinated limb motion, and interaction with obstacles or objects. The official project also provides state, visual, and tactile observation interfaces; these capabilities describe the full benchmark.

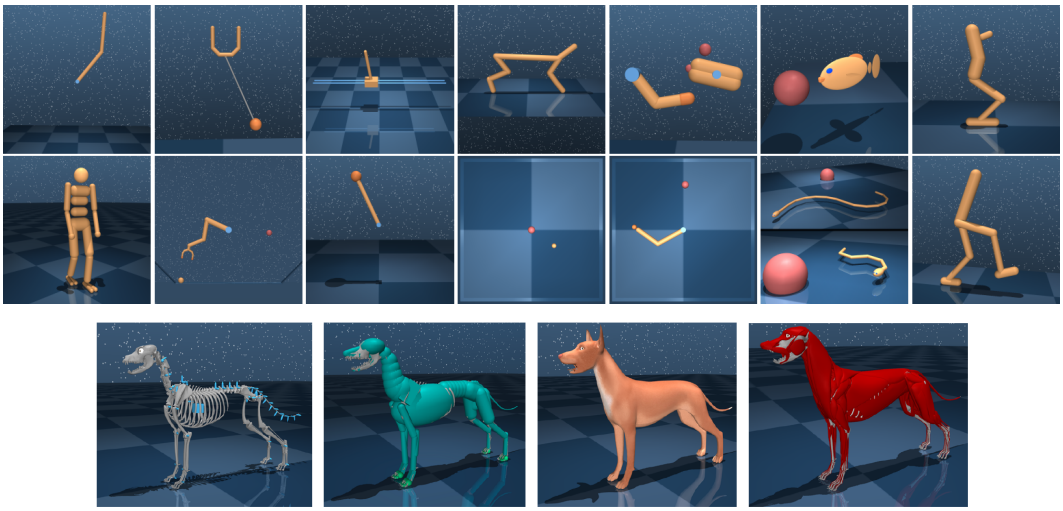


Figure 7: DeepMind Control Suite examples. Top: the domain overview from the official `dm_control` repository. Bottom: the dog model's skeleton, collision geometry, visual skin, and muscle visualization, reproduced from Figure 6 of Tunyasuvunakool et al. (2020). The panels illustrate the suite and its bodies; the evaluated task collections are specified in Appendix B.2.

Following BRC, our HB-Medium setting uses the Unitree H1 configuration without dexterous hands and nine locomotion tasks: `walk`, `stand`, `run`, `stair`, `crawl`, `pole`, `slide`, `hurdle`, and `maze`. The corresponding environment identifiers have the form `h1-{task}-v0`. Together, these tasks cover posture control, level-ground locomotion, and movement through structured terrain or obstacles. We train for 1M interactions per task and report mean normalized return. Figure 8 provides visual context from the official project. HB-Medium is used in the main comparison and is not part of the 165-task capacity-scaling experiment.

B.4 ShadowHand

The dexterous-manipulation suite introduced by Huang et al. (2021) uses a simulated Shadow Hand to manipulate objects with diverse geometries. The released collection contains 114 objects drawn from the YCB and ContactDB collections (Calli et al., 2015; Brahmbhatt et al., 2019). Different shapes change the available contacts and the finger motions needed to manipulate an object. The original work also studies generalization to held-out objects using geometry-based representations.

Our capacity-scaling experiment uses the 85 training objects in BRC's ShadowHand collection, with sparse task rewards. Each object defines a task and receives 1M training interactions. This is a joint training experiment on the selected objects; it does not evaluate the original paper's held-out-object generalization protocol. ShadowHand contributes 85 of the 165 jointly trained tasks, alongside 50 MT50 tasks and 30 DMC tasks. We report its group-average score separately from those of MT50 and DMC. Figure 9 illustrates the original suite, and Table 5 lists the selected objects.

B.5 Task collections

The following tables identify the task collections used by the BRC protocol followed in our experiments (Nauman et al., 2026). Names identify tasks independently of package versions and preprocessing; those implementation settings are recorded in Table 7.

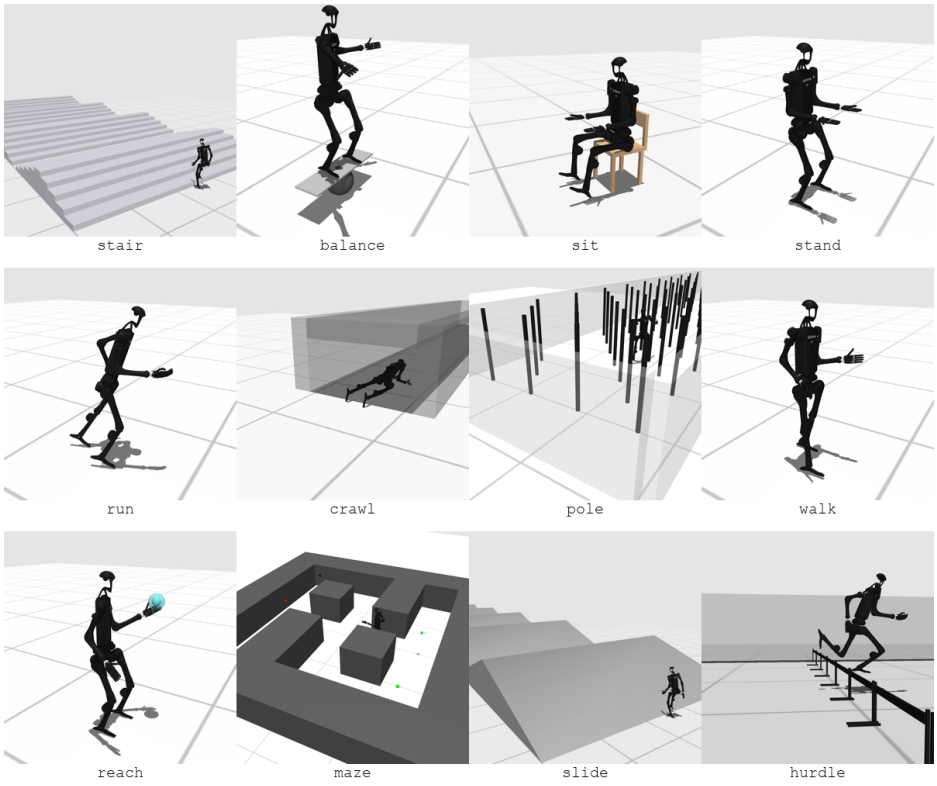


Figure 8: Locomotion-related environments from the official HumanoidBench project (Sferrazza et al., 2024). This original project montage also includes `balance`, `sit`, and `reach`, which are outside our nine-task HB-Medium subset, and depicts a hand-equipped robot. Our evaluated subset uses the H1 configuration without dexterous hands. Image reproduced under CC BY-SA 4.0.

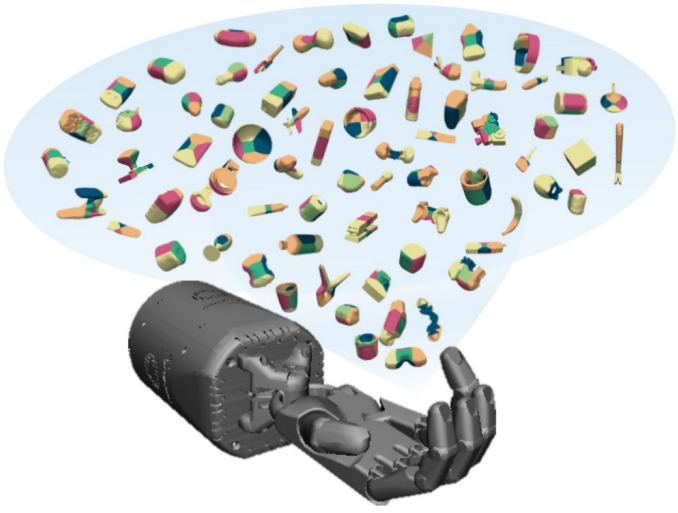


Figure 9: The Shadow Hand and the object diversity of the original manipulation suite. Reproduced from Figure 1 of Huang et al. (2021). The image illustrates the full source collection; our experiment uses its 85-object training subset.

Table 3: The 50 MT50 task names.

assembly	hand-insert	plate-slide-back-side
basketball	drawer-close	peg-insert-side
bin-picking	drawer-open	peg-unplug-side
box-close	faucet-open	soccer
button-press-topdown	faucet-close	stick-push
button-press-topdown-wall	hammer	stick-pull
button-press	handle-press-side	push
button-press-wall	handle-press	push-wall
coffee-button	handle-pull-side	push-back
coffee-pull	handle-pull	reach
coffee-push	lever-pull	reach-wall
dial-turn	pick-place-wall	shelf-place
disassemble	pick-out-of-hole	sweep-into
door-close	pick-place	sweep
door-lock	plate-slide	window-open
door-open	plate-slide-side	window-close
door-unlock	plate-slide-back	

Table 4: The 30 DMC tasks in joint capacity scaling, grouped by domain. Task suffixes retain the naming of the reference task collection.

Domain	Count	Task suffixes
Walker	5	stand, walk, run, walk_backwards, run_backwards
Cheetah	5	run, run_backwards, run_front, run_back, jump
Reacher	4	easy, hard, three_easy, three_hard
Acrobot	1	swingup
Pendulum	2	swingup, spin
Cartpole	4	balance, balance_sparse, swingup, swingup_sparse
Ball-in-cup	2	catch, spin
Finger	3	spin, turn_easy, turn_hard
Fish	1	swim
Hopper	3	stand, hop, hop_backwards

Table 5: The 85 ShadowHand training-object identifiers in the joint experiment.

e_toy_airplane	d_toy_airplane	d_cups
knife	wood_block	camera
flat_screwdriver	small_marker	d_lego_duplo
elephant	sugar_box	golf_ball
apple	ball	k_toy_airplane
scissors	torus	b_cups
i_cups	i_toy_airplane	softball
cup	chain	wine_glass
foam_brick	j_cups	chips_can
pudding_box	c_toy_airplane	cube
wristwatch	airplane	master_chef_can
padlock	nine_hole_peg_test	alarm_clock
power_drill	water_bottle	gelatin_box
binoculars	c_cups	h_lego_duplo
b_lego_duplo	medium_clamp	baseball
ps_controller	large_marker	light_bulb
mouse	h_cups	banana
hammer	b_colored_wood_blocks	rubber_duck
f_lego_duplo	j_lego_duplo	headphones
piggy_bank	f_toy_airplane	i_lego_duplo
can	toothbrush	b_toy_airplane
extra_large_clamp	tennis_ball	pitcher_base
peach	mug	j_toy_airplane
a_lego_duplo	sponge	g_lego_duplo
racquetball	k_lego_duplo	cracker_box
tuna_fish_can	phillips_screwdriver	orange
a_cups	f_cups	e_cups
pan	c_lego_duplo	
strawberry	d_marbles	

B.6 BASELINES

The main comparison includes nine baselines within the BRC training pipeline. BRC-FP32 provides the high-precision reference. The remaining methods are adapted to the FP8 persistent-state setting. The descriptions below summarize the source methods. Shared training and evaluation settings are given in Appendix B.8.

BRC-FP32. Bigger, Regularized, Categorical (BRC) combines high-capacity value networks, regularization, categorical temporal-difference learning, and learned task embeddings for online multi-task reinforcement learning (Nauman et al., 2026). Its critic uses a categorical value representation trained with cross-entropy. BRC-FP32 supplies both the learning framework for our precision comparisons and the high-precision reference for evaluating performance and training costs. It also serves as the reference in the capacity-scaling experiment.

Independent stochastic rounding (SR). SR rounds a value to one of its two adjacent representable numbers with probabilities determined by their distances from that value (Croci et al., 2022). For an in-range value on a fixed grid, the rounded result equals the input in expectation, allowing small updates to contribute in expectation. Independent SR samples these decisions separately across elements and provides the direct reference for assessing the benefit of coordinating rounding errors.

FP16RL. FP16RL develops numerical modifications that enable soft actor–critic training in half precision (Björck et al., 2021). These include stable reformulations of sensitive computations, loss scaling, and an Adam update that stores the square root of the second moment. Kahan summation compensates for lost increments in critic parameter updates, entropy-temperature updates, and target-network averaging. The method therefore provides a low-precision RL reference based on numerical stabilization and explicit compensation. Its original training format is FP16.

ECO. The Error-Compensating Optimizer (ECO) updates quantized parameters without maintaining a separate full-precision master-weight copy (Nikdan et al., 2026). It feeds the residual from parameter quantization into the optimizer's existing momentum, so information lost during writeback can influence later updates. The source work develops this approach for momentum SGD and Adam. ECO represents error compensation through optimizer dynamics, without adding a separate parameter-sized residual buffer.

FlashOptim. FlashOptim compresses training state through weight splitting and optimizer-state quantization (Ortiz et al., 2026). A low-precision weight and an integer correction term jointly represent a more precise parameter, which is reconstructed during the optimizer update. The correction is scaled to the local spacing of the weight representation. The method also compresses optimizer moments to eight bits and fuses their decoding, update, and encoding. It provides a reference that retains update information through explicit residual storage.

MS-AMP O2 (FP8-LM). FP8-LM introduces a staged mixed-precision training framework that extends FP8 to gradients, communication, and optimizer state (Peng et al., 2023). Its precision-decoupling strategy assigns different formats according to the numerical sensitivity of each component, while scaling helps manage the limited FP8 range. In the documented MS-AMP O2 configuration, the first optimizer moment uses FP8, the second uses FP16, and weights remain FP16. This is a reference for selective precision allocation in the optimizer.

COAT. COAT reduces training memory through FP8 optimizer-state compression and activation compression (Xi et al., 2025). For optimizer states, dynamic range expansion applies a group-dependent nonlinear transform before quantization to improve use of the available FP8 values. For activations, COAT combines different quantization granularities across linear and nonlinear operations. Its optimizer-state component provides a reference for improving numerical representation before quantization.

Correlated Quantization (CQ). CQ was introduced for distributed mean estimation and optimization (Suresh et al., 2022). It coordinates randomness across clients while preserving unbiased quantization, reducing the error of the aggregate estimate when client values are similar. Its analysis

relates estimation error to the dispersion of the inputs. CQ provides a rounding reference that uses dependence between errors to improve their aggregate behavior.

Signed-SR$_\epsilon$. This rule biases SR toward a selected direction (Xia et al., 2022). In its original gradient-descent setting, the gradient sign determines the preferred rounding direction, and the bias parameter controls the probability shift, clipped to valid probabilities. The resulting bias favors the descent direction and can help prevent stagnation. This comparator examines deliberately biased rounding as a way to retain useful update information.

Round-to-nearest (RTN) ablation reference. RTN selects the nearest representable value (Croci et al., 2022); an update smaller than half the local spacing in its direction can therefore round back to the stored value.

B.7 Baseline Precision Configurations

Table 6 summarizes the implemented baseline configurations. The main comparison uses a common FP8 main-weight storage setting while retaining the method-specific state representations listed below.

Table 6: Precision configurations of the implemented baselines. Formats refer to residual Dense weights and their optimizer states. FP8 denotes E4M3; A, C, and T denote the actor, online critic, and target critic. Targets store absolute parameters.

Method	Online weights / rounding	Adam states	Target / rounding	Extra state
BRC-FP32	FP32	FP32	FP32	None
Independent SR	FP8 / SR	FP8 / SR	FP8 / SR	None
FP16RL	FP8 / RTN	FP8 $(m, \sqrt{v})$	FP8 / RTN	FP16 Kahan (A/C/T)
ECO	FP8 / RTN	FP8 (m, v)	FP8 / RTN	None
FlashOptim	FP8 / RTN	m: INT8 v: UINT8	FP8 / RTN	INT8 weight corrections (A/C)
MS-AMP O2 (FP8-LM)	FP8 / RTN	FP8 (m, v)	FP8 / RTN	None
COAT	FP8 / RTN	FP8 (m, v), DRE	FP8 / RTN	FP32 DRE metadata
Correlated Quantization	FP8 / CQ (2×2 groups)	FP8 (m, v)	FP8 / RTN	None
Signed-SR$_\epsilon$	FP8 / signed-SR ($\epsilon = 0.05$)	FP8 (m, v)	FP8 / RTN	None

FP16RL additionally uses FP32 temperature compensation and initial loss/target scales of 10^4. ECO feeds weight error into momentum before moment quantization. FlashOptim applies softsign and square-root transforms to its first- and second-moment codes. MS-AMP O2 uses power-of-two online-weight and moment scales and transient FP8 gradient quantization. COAT uses dynamic range expansion (DRE), storing an FP32 exponent and center per moment block in addition to its scale, and quantizes residual normalization/ReLU activations in groups of 16. CQ uses stratified sampling within 2×2 weight groups; signed-SR uses the sign of the candidate weight displacement.

B.8 Training and evaluation protocol

All experiments use five random seeds. In the main comparison, methods use matched architectures, batch sizes, and update frequencies within each benchmark. Evaluation uses 10 episodes per task every 25K steps. Let $s_{j,k}$ denote the score computed from these episodes for task k under training seed j, using success rate for MT50 and the benchmark-specific score transformation for return-based metrics. For a collection of T tasks, aggregation is

$$\bar{s}_j = \frac{1}{T}\sum_{k=1}^{T} s_{j,k}, \qquad \widehat{\mu} = \frac{1}{5}\sum_{j=1}^{5} \bar{s}_j. \tag{16}$$

The 95% confidence intervals are obtained by bootstrapping the five seed-level task averages. For joint training, each benchmark group has its own task average. MT50 success is expressed as a fraction in the main-result table and as a percentage in the ablation table, matching the corresponding main-text presentations.

The training and evaluation settings are recorded in Tables 7–10. The capacity experiment compares BRC-FP32 with critic width 4096 and CoSR with critic width 8192, at approximately matched peak JAX-allocated training memory. Each receives the same per-task interaction budget. The memory constraint concerns peak allocated training memory; wall-clock cost is measured separately. Shared entries in these tables use one actor and one two-critic BroNet (Nauman et al., 2024) with ReLU and LayerNorm (Ba et al., 2016). The actor is a tanh-Gaussian. Observations are zero-padded on the right, and actions are masked to each task's own dimension. A true terminal blocks the value bootstrap, while truncation does not. Meta-World resamples the goal at every reset. The reward scale uses the per-task empirical return and empirical entropy. Temperature is optimized by Adam with $\beta_1 = 0.5$; its target is $-d_a/2$ on the padded action dimension. DMC action repeats are the suite physics substeps, and the HB repeat is the environment frame skip. Joint-165 horizons and repeats are listed in the order MT50, DMC, and ShadowHand. DMC-Hard trains the four dog tasks and the three humanoid tasks as separate agents and merges the scores. Joint-165 parameter counts list the shared actor and then the critics at widths 4096 and 8192.

Table 7: Environment and data-collection settings. Joint-165 denotes joint training on MT50, DMC-30, and ShadowHand-85.

Setting	MT50	DMC-Hard	HB-Medium	Joint-165
Number of tasks	50	7	9	165
Environment steps per task	500K	500K	1M	1M
Training seeds	5	5	5	5
Evaluation episodes per task	10	10	10	10
Evaluation interval (steps)	25K	25K	25K	25K
Step-counter unit for this interval	Environment steps per task			
Environment package version / commit	Meta-World 2.0.0	dm_control 1.0.44	HB `cb11890`; Gym 0.29.1; MuJoCo 3.1.6	MW 2.0.0; DMC 1.0.44; Robotics 1.3.1
Observation fields and task encoding	39-D; L1-32	Suite fields; L1-32	51-D; L1-32	68-D padded; L1-32
Observation and action dimensions	39 / 4	Dog 223/38; hum. 67/21	51 / 19	68 / 20
Observation normalization	Identity			
Action scaling / padding / masks	$[-1, 1]$, then mask			
Episode horizon	200	1,000	1,000	200 / 1,000 / 100
Action repeat	1	Dog 3, hum. 5	10	1 / suite / 1
Termination and truncation handling	Terminal blocks bootstrap			
Environments per task	1			
Task sampling / replay balancing	Uniform, synced			
Random exploration / warmup steps	5,000			
Replay-buffer capacity per task	1M			
Batch size	1,024			
Updates per collection step	2			
Transitions per collection step	50	Dog 4, hum. 3	9	165
Training reward transformation	Return scale, $v_{\max} = 10$			

Table 8: Architecture and optimization settings. The two widths shown for Joint-165 correspond to BRC-FP32 and CoSR, respectively.

Setting	MT50	DMC-Hard	HB-Medium	Joint-165
Optimizer	AdamW	AdamW	AdamW	AdamW
Actor architecture / depth / width	BroNet, 1×256			
Critic architecture / depth	BroNet, depth 2			
Critic width	4096	4096	4096	4096 / 8192
Number of critics	2			
Actor / critic parameter counts	153,608 / 135,784,202	Dog 218,188 / 137,568,586; hum. 169,514 / 136,151,338	164,390 / 136,004,074	169,256 / 136,156,522 / 540,743,018
Task-embedding dimension	32, L1-normalized			
Categorical critic bins and support	101 bins on $[-10, 10]$			
Discount factor γ	0.99			
Actor learning rate	3×10^{-4}			
Critic learning rate	3×10^{-4}			
Temperature learning rate	3×10^{-4}			
Adam β_1, β_2, and ϵ	0.9, 0.999, 10^{-8}			
Weight decay λ	10^{-4}			
Learning-rate schedule	Constant			
Actor update frequency	Every critic update			
Target update frequency	Every critic update			
Target smoothing coefficient τ	0.005			
Entropy target and initial temperature	$-d_a/2$; 0.1			
Gradient clipping and other regularizers	None			

Table 9: Precision and storage specification for CoSR. The format of persistent state is distinguished from computation and accumulation precision.

Component	Setting
Main actor and critic weight matrices	FP8 codes, separate scales
Main Adam first- and second-moment states	FP8 codes; one scale per moment
Target representation	FP8 lag relative to the online critic
Dense compensation for these states	None
Weight FP8 encoding variant	E4M3 residuals; one FP32 scale per matrix; CoSR on non-overlapping 2×2 blocks. Input, policy-head, and 101-bin kernels are BF16
First- and second-moment FP8 variants	Actor and critic residual moments: E4M3, separate block-128 FP32 scales. CoSR couples adjacent coordinates and complements m and v
Target-lag FP8 variant and rounding rule	E4M3, one FP32 scale per matrix, round-to-nearest. Target $=$ online weight $+$ dequantized lag
Scale dtype, granularity, and update rule	FP32. Weight and lag: $\mathrm{amax}/448$. Moments: block-128 amax. Backward: delayed amax history of length 1,024
Clipping, saturation, and subnormal handling	E4M3 range ± 448. Zero amax uses scale 1. Subnormal moment scales round up to the next normal
Biases, normalization parameters, and embeddings	FP32
Temperature and other scalar states	FP32
Forward / backward operand precision	Forward activations and residual weights: E4M3. Backward output gradients: E5M2
Rounding random-number construction	`PRNGKey(seed)`, update step, and path CRC32. Not stored

Table 10: Evaluation and measurement details shared by the reported comparisons, with benchmark-specific entries where applicable.

Item	Setting
GPU model	RTX PRO 6000 Blackwell
Number of GPUs per run	1
CPU and host memory	2× Xeon Platinum 8457C; 503 GiB
Driver, CUDA, JAX, and compiler versions	580.95.05; CUDA 12.8; JAX 0.6.0; XLA autotune 0
Training and evaluation seed identifiers	1, 2, 3, 42, 123
Evaluation action selection	$\tanh(\mathrm{mean})$, temperature 0
Final checkpoint or averaging window	Last evaluation
DMC return-normalization rule	Return$/1{,}000$
HB return-normalization rule and reference scores	$(R - R_{\mathrm{rand}})/(R_\star - R_{\mathrm{rand}})$. $R_\star/R_{\mathrm{rand}}$ in Appendix B.3 order: 700/2.377, 800/10.545, 700/2.02, 700/3.112, 700/272.658, 700/20.09, 700/3.191, 700/2.214, 1,200/106.441
ShadowHand group-score definition	Mean success, 85 objects
Task weighting	Equal within each group
Uncertainty unit	Seed-level task average
Confidence level	95%
Bootstrap method and resample count	Percentile, 50,000
Persistent-state accounting	Weights, moments, targets, scales, compensation
Peak-memory accounting	Training state and temporaries
Replay-buffer location and memory inclusion	CPU NumPy; excluded from GPU totals
Device-memory measurement API and units	NVML, 20 ms; GiB $=$ bytes$/1024^3$; JAX peak separate
Update timing	After warmup; excludes compile, interaction, evaluation
Warmup and timed-update counts	100 calls in 6,000–6,099, 12,000–12,099, 19,901–20,000; 10 warmups each
Latency statistic and repetition count	Mean, median, P95; 300 calls, divided by 2

C PROOFS OF THE THEORETICAL RESULTS

C.1 CONDITIONAL ROUNDING MODEL AND AN EXTREMAL-COUPLING LEMMA

All expectations in the rounding proofs are conditional on the current candidates, quantization grids (including their scales), and recorded operands or sensitivity coefficients. These quantities are fixed before fresh rounding randomness is generated. We use the ideal scalar SR model on finite bracketing grid points; candidates are within the representable range, so no additional clipping changes their SR marginals. The grid spacing need not be uniform or shared across coordinates.

For a scalar candidate z_i bracketed by ℓ_i and $\ell_i + h_i$, write

$$z_i = \ell_i + h_i p_i, \qquad \widehat{z}_i = \ell_i + h_i B_i, \qquad B_i \sim \mathrm{Bernoulli}(p_i). \tag{17}$$

Here $h_i > 0$ and $p_i \in [0, 1]$. Exactly representable candidates are rounded deterministically and have zero error; they can also be treated as degenerate coordinates with $h_i = 0$ and $p_i = 0$. For $e_i = \widehat{z}_i - z_i = h_i(B_i - p_i)$,

$$\mathbb{E}[e_i] = 0, \qquad \nu_i := \mathbb{E}[e_i^2] = h_i^2 p_i(1 - p_i), \qquad \mathrm{Cov}(e_i, e_j) = h_i h_j \,\mathrm{Cov}(B_i, B_j). \tag{18}$$

Thus fixing the scalar SR marginals fixes all diagonal second moments, regardless of dependence between coordinates. In particular, $\mathbb{E}\|e\|_2^2 = \sum_i \nu_i$ is unchanged by the coupling.

Lemma: extremal scalar coupling. Let $B_1 \sim \mathrm{Bernoulli}(p)$ and $B_2 \sim \mathrm{Bernoulli}(q)$. Every coupling of these marginals satisfies

$$\kappa_-(p, q) \le \mathrm{Cov}(B_1, B_2) \le \kappa_+(p, q), \tag{19}$$

where

$$\begin{aligned} \kappa_-(p, q) &= \max(0, p + q - 1) - pq \le 0, \\ \kappa_+(p, q) &= \min(p, q) - pq \ge 0. \end{aligned} \tag{20}$$

For $U \sim \mathcal{U}(0, 1)$, the common-uniform construction

$$B_1 = \mathbf{1}[U < p], \qquad B_2 = \mathbf{1}[U < q] \tag{21}$$

attains κ_+, and the complementary-uniform construction

$$B_1 = \mathbf{1}[U < p], \qquad B_2 = \mathbf{1}[1 - U < q] \tag{22}$$

attains κ_-. Both constructions preserve the scalar marginals.

Proof. Set $q_{11} = \Pr(B_1 = 1, B_2 = 1)$. Nonnegativity of the four joint probabilities gives

$$q_{11} \ge 0, \quad p - q_{11} \ge 0, \quad q - q_{11} \ge 0, \quad 1 - p - q + q_{11} \ge 0. \tag{23}$$

Consequently, $\max(0, p+q-1) \le q_{11} \le \min(p, q)$. Since $\mathrm{Cov}(B_1, B_2) = q_{11} - pq$, this proves the bounds. In the common-uniform construction, the two upward-rounding events intersect on an interval of length $\min(p, q)$. In the complementary construction, their intersection is $\{1 - q < U < p\}$, of length $\max(0, p + q - 1)$. Finally, U and $1 - U$ are both uniform, so each marginal is unchanged. □

A consequence used below is that, for fixed scalar SR marginals and fixed coefficients a, b,

$$\mathbb{E}(ae_1 + be_2)^2 = a^2\nu_1 + b^2\nu_2 + 2ab\,\mathrm{Cov}(e_1, e_2). \tag{24}$$

Its minimum is attained by complementary uniforms when $ab > 0$, and by common uniforms when $ab < 0$. If $ab = 0$, the coupling does not affect the objective. This is a minimum over all couplings of the fixed scalar marginals, not merely over the two displayed constructions.

C.2 PROOF OF PROPOSITION 1: BIDIRECTIONALLY OPTIMAL WEIGHT COUPLING

Use zero-based indices. Fix the recorded forward input $X \in \mathbb{R}^{B \times d_{\mathrm{in}}}$, output cotangent $G \in \mathbb{R}^{B \times d_{\mathrm{out}}}$, candidate weights $Z \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}$, and all scalar quantization grids, where B is the batch size. Let $\widehat{W}$ be the weights after writeback and $E = \widehat{W} - Z$ their rounding error. The forward and backward-input risks are

$$\mathcal{R}_f(P) = \mathbb{E}_P\|XE\|_F^2, \qquad \mathcal{R}_b(P) = \mathbb{E}_P\|GE^\top\|_F^2, \tag{25}$$

where P is the joint rounding distribution and $\|\cdot\|_F$ is the Frobenius norm. Let h_{ij} and p_{ij} be the grid width and upward-rounding probability at the global matrix coordinate (i, j), and write

$$\nu_{ij} = h_{ij}^2 p_{ij}(1 - p_{ij}). \tag{26}$$

Pair adjacent rows and columns to form the fixed partition

$$\begin{aligned} \mathcal{I}_r &= \{2r, 2r+1\} \cap \{0, \dots, d_{\text{in}} - 1\}, \\ \mathcal{J}_c &= \{2c, 2c+1\} \cap \{0, \dots, d_{\text{out}} - 1\}, \\ \mathcal{B}_{rc} &= \mathcal{I}_r \times \mathcal{J}_c, \end{aligned} \tag{27}$$

with $0 \le r < \lceil d_{\text{in}}/2 \rceil$ and $0 \le c < \lceil d_{\text{out}}/2 \rceil$. When both dimensions are even, all blocks are the complete 2×2 blocks used in the main-text construction. For an odd dimension, the partition also contains 2×1, 1×2, or 1×1 boundary blocks. We specify their rounding below to extend the proof to these cases. Let $\mathcal{P}_2$ contain all distributions that preserve the scalar SR marginals and are independent across the blocks of this partition. Dependence within a block is otherwise unrestricted.

Step 1: isolate the coupling-dependent terms. Define $K_X = X^\top X$ and $K_G = G^\top G$. Expanding the forward risk under an arbitrary $P \in \mathcal{P}_2$ gives

$$\begin{aligned} \mathcal{R}_f(P) &= \sum_{j=0}^{d_{\text{out}}-1} \sum_{i,k=0}^{d_{\text{in}}-1} (K_X)_{ik}\, \mathbb{E}_P[E_{ij}E_{kj}] \\ &= D_f + 2 \sum_{r=0}^{\lfloor d_{\text{in}}/2 \rfloor - 1} \sum_{j=0}^{d_{\text{out}}-1} a_r \operatorname{Cov}_P(E_{2r,j}, E_{2r+1,j}), \end{aligned} \tag{28}$$

where

$$D_f = \sum_{i,j} (K_X)_{ii} \nu_{ij}, \qquad a_r = (K_X)_{2r,2r+1} = \langle X_{:,2r}, X_{:,2r+1} \rangle. \tag{29}$$

The term D_f is fixed by the scalar marginals. For a fixed column, distinct rows that are not in the same row pair belong to different blocks; their errors are independent and zero-mean, so their cross term vanishes. Only the vertical within-block pairs remain.

Likewise,

$$\begin{aligned} \mathcal{R}_b(P) &= \sum_{i=0}^{d_{\text{in}}-1} \sum_{j,k=0}^{d_{\text{out}}-1} (K_G)_{jk}\, \mathbb{E}_P[E_{ij}E_{ik}] \\ &= D_b + 2 \sum_{i=0}^{d_{\text{in}}-1} \sum_{c=0}^{\lfloor d_{\text{out}}/2 \rfloor - 1} b_c \operatorname{Cov}_P(E_{i,2c}, E_{i,2c+1}), \end{aligned} \tag{30}$$

where

$$D_b = \sum_{i,j} (K_G)_{jj} \nu_{ij}, \qquad b_c = (K_G)_{2c,2c+1} = \langle G_{:,2c}, G_{:,2c+1} \rangle. \tag{31}$$

The sums in D_f, D_b run over all matrix coordinates; sums over nonexistent row or column pairs are empty. Only the horizontal within-block pairs remain in the backward risk. Covariances between positions on a block diagonal occur in neither decomposition, because such positions share neither a row nor a column.

Step 2: lower-bound every relevant pairwise contribution. For compactness, define

$$\kappa^{[a]}(p, q) = \begin{cases} \kappa_-(p, q), & a \ge 0, \\ \kappa_+(p, q), & a < 0. \end{cases} \tag{32}$$

For any two coordinates with errors e_i, e_j, the coupling lemma yields

$$a \operatorname{Cov}_P(e_i, e_j) \ge a h_i h_j \kappa^{[a]}(p_i, p_j). \tag{33}$$

For $a > 0$ this uses the lower covariance bound; for $a < 0$ it uses the upper bound and reverses the inequality upon multiplication. For $a = 0$ both sides are zero. Applying this bound with the same summation ranges as in equation 28 and equation 30 yields, for every $P \in \mathcal{P}_2$,

$$\mathcal{R}_f(P) \geq D_f + 2 \sum_{r,j} a_r h_{2r,j} h_{2r+1,j} \kappa^{[a_r]}(p_{2r,j}, p_{2r+1,j}) =: L_f, \tag{34}$$

$$\mathcal{R}_b(P) \geq D_b + 2 \sum_{i,c} b_c h_{i,2c} h_{i,2c+1} \kappa^{[b_c]}(p_{i,2c}, p_{i,2c+1}) =: L_b. \tag{35}$$

It remains to show that a single feasible joint distribution attains both bounds. Pairwise extrema need not be jointly compatible in a general coupling problem; the CoSR construction supplies the required compatibility here.

Step 3: the XOR construction attains all required extrema. In each block draw an independent $U_{rc} \sim \mathcal{U}(0,1)$. For complete row and column pairs use the main-text flip flags; for a singleton boundary row or column set the corresponding flag to zero:

$$\begin{aligned} f_r &= \begin{cases} \mathbf{1}[a_r \geq 0], & 2r+1 < d_{\text{in}}, \\ 0, & \text{otherwise}, \end{cases} \\ k_c &= \begin{cases} \mathbf{1}[b_c \geq 0], & 2c+1 < d_{\text{out}}, \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{36}$$

For local indices $i, j \in \{0,1\}$ with $(2r+i, 2c+j) \in \mathcal{B}_{rc}$, define

$$\phi_{ij}^{(r,c)} = (i f_r) \oplus (j k_c), \tag{37}$$

where $\oplus$ denotes exclusive OR on bits, and assign

$$V_{2r+i,2c+j} = \begin{cases} U_{rc}, & \phi_{ij}^{(r,c)} = 0, \\ 1 - U_{rc}, & \phi_{ij}^{(r,c)} = 1. \end{cases} \tag{38}$$

The upward-rounding indicator at each global coordinate is

$$B_{2r+i,2c+j} = \mathbf{1}[V_{2r+i,2c+j} < p_{2r+i,2c+j}]\,. \tag{39}$$

All assigned uniforms have the $\mathcal{U}(0,1)$ marginal, so each coordinate retains its prescribed scalar SR distribution. The independent draws U_{rc} also give independence across blocks. Thus the resulting distribution $P^\star$ belongs to $\mathcal{P}_2$.

Whenever a block contains two rows, each column present in that block satisfies

$$\phi_{0j}^{(r,c)} \oplus \phi_{1j}^{(r,c)} = f_r. \tag{40}$$

Hence vertical pairs use complementary uniforms if $a_r \geq 0$ and common uniforms if $a_r < 0$. The coupling lemma shows that every term in equation 34 attains equality. Similarly, whenever a block contains two columns, each row present satisfies

$$\phi_{i0}^{(r,c)} \oplus \phi_{i1}^{(r,c)} = k_c, \tag{41}$$

so every horizontal pair attains equality in equation 35. In a complete 2×2 block, these choices coexist because the two vertical edges share f_r and the two horizontal edges share k_c; the same XOR assignment satisfies all four edges. A 2×1 or 1×2 boundary block has a single relevant edge, which is optimized by its row or column flag. A 1×1 block uses scalar SR and contributes only fixed diagonal terms. Consequently,

$$\boxed{\mathcal{R}_f(P^\star) = L_f = \min_{P \in \mathcal{P}_2} \mathcal{R}_f(P), \qquad \mathcal{R}_b(P^\star) = L_b = \min_{P \in \mathcal{P}_2} \mathcal{R}_b(P).} \tag{42}$$

This proves Proposition 1 for complete blocks and the stated boundary extension. □

Comparison with independent SR. Under independent SR, all off-diagonal error covariances vanish, so $\mathcal{R}_f(P_{\text{ind}}) = D_f$ and $\mathcal{R}_b(P_{\text{ind}}) = D_b$. Every correction term in $L_f - D_f$ and $L_b - D_b$ is nonpositive. The claimed inequalities against independent SR follow immediately. A given risk is strictly reduced whenever it contains a pair with a nonzero channel inner product, positive grid widths, and both upward probabilities strictly between zero and one. Otherwise some or all pairwise gains may be zero. This comparison concerns the fixed recorded operands and the specified coupling class.

C.3 Proof of Proposition 2: update-aware coupling of Adam moments

Fix one coordinate of the candidate first and second moments M, V, with $V > 0$, at update $t+1$. For the moment decay rates $\beta_1, \beta_2 \in [0, 1)$ and $t \geq 0$, abbreviate the main-text bias-correction factors as

$$\begin{aligned} c_1 &:= c_{1,t+1} = 1 - \beta_1^{t+1} > 0, \\ c_2 &:= c_{2,t+1} = 1 - \beta_2^{t+1} > 0. \end{aligned} \tag{43}$$

The bias-corrected update direction in the main text is

$$F(M, V) = \frac{M/c_1}{\sqrt{V/c_2} + \epsilon}, \qquad \epsilon > 0. \tag{44}$$

Writing $\rho = \sqrt{V/c_2}$, its derivatives are

$$\alpha = \frac{\partial F}{\partial M} = \frac{1}{c_1(\rho + \epsilon)} > 0, \qquad \beta = \frac{\partial F}{\partial V} = -\frac{M}{2c_1c_2\rho(\rho + \epsilon)^2}. \tag{45}$$

Thus $\operatorname{sign}(\beta) = -\operatorname{sign}(M)$. Keeping the bias corrections explicitly shows that they do not change the sign rule used by CoSR.

Let h_m, h_v and p_m, p_v denote the respective grid widths and upward probabilities. Under any coupling of the fixed SR marginals,

$$e_m = h_m(B_m - p_m), \qquad e_v = h_v(B_v - p_v). \tag{46}$$

The linearized direction error is

$$\delta u_{\text{lin}} = \alpha e_m + \beta e_v. \tag{47}$$

Its objective decomposes as

$$\begin{aligned} J(P) &:= \mathbb{E}_P(\delta u_{\text{lin}})^2 \\ &= \alpha^2 h_m^2 p_m(1 - p_m) + \beta^2 h_v^2 p_v(1 - p_v) + 2\alpha\beta h_m h_v \operatorname{Cov}_P(B_m, B_v). \end{aligned} \tag{48}$$

Only the final covariance term depends on the coupling.

If $M > 0$, then $\alpha\beta < 0$, so minimizing J requires the largest possible covariance, namely $\kappa_+(p_m, p_v)$. This is attained by $U_v = U_m$. If $M < 0$, then $\alpha\beta > 0$, so minimizing J requires $\kappa_-(p_m, p_v)$, attained by $U_v = 1 - U_m$. If $M = 0$, then $\beta = 0$ and every coupling has the same linearized objective; choosing $U_v = U_m$ is therefore also optimal. These cases give exactly the rule

$$\boxed{U_v = \begin{cases} U_m, & M \geq 0, \\ 1 - U_m, & M < 0. \end{cases}} \tag{49}$$

By the extremal-coupling lemma, this rule minimizes J over all joint laws of (B_m, B_v) with the prescribed marginals. In particular,

$$\boxed{\mathbb{E}(\delta u_{\text{CoSR,lin}})^2 \leq \mathbb{E}(\delta u_{\text{SR,lin}})^2.} \tag{50}$$

This proves Proposition 2. □

Spatial coupling and scope of the result. The argument requires each U_m to be marginally uniform, not independent of uniforms assigned to other coordinates. The spatial XOR construction preserves this marginal and can therefore coexist with the optimal same-coordinate m/v rule. Moreover, the squared Euclidean norm of the linearized update error is the sum of its coordinatewise squares, so summing the coordinatewise inequalities preserves the comparison. This does not establish an optimum for additional cross-coordinate operator-weighted objectives or for the exact nonlinear Adam ratio.

The linearization is defined at $V > 0$. At $V = 0$, zero is represented exactly and $e_v = 0$, so there is no nontrivial second-moment rounding to couple; the remaining perturbation is determined by the first-moment marginal. No derivative of $\sqrt{V}$ at zero is used. Finally, as specified in the main text, the current parameter update is computed before the new moments are quantized. The proposition describes the reconstruction error of stored moment states under the direction map F, not a change to the already-computed current update.

C.4 EQUIVALENCE OF THE RELATIVE LAG RECURRENCE AND TARGET SOFT UPDATES

Let θ_t and θ_{t+1} be the actual online parameters before and after weight writeback. Let $\widehat{L}_t$ be the decoded stored lag, so the current reconstructed target is

$$\widehat{\bar{\theta}}_t = \theta_t + \widehat{L}_t. \tag{51}$$

Define the actual online displacement

$$\Delta\theta_t = \theta_{t+1} - \theta_t. \tag{52}$$

Consider the ordinary soft update applied to the currently represented target:

$$\bar{\theta}^+_{t+1} = (1-\tau)\widehat{\bar{\theta}}_t + \tau\theta_{t+1}, \qquad 0 < \tau \le 1. \tag{53}$$

Its lag relative to the new online parameters is

$$\begin{aligned} L^+_{t+1} &= \bar{\theta}^+_{t+1} - \theta_{t+1} \\ &= (1-\tau)(\theta_t + \widehat{L}_t) + \tau\theta_{t+1} - \theta_{t+1} \\ &= (1-\tau)\big(\widehat{L}_t - (\theta_{t+1} - \theta_t)\big) \\ &= \boxed{(1-\tau)(\widehat{L}_t - \Delta\theta_t).} \end{aligned} \tag{54}$$

Thus, before lag quantization, the relative-state update is algebraically identical to the original soft update.

Let Q_L denote quantization to the lag's scale-defined FP8 grid followed by decoding to parameter units. Define the lag writeback error by

$$\widehat{L}_{t+1} = Q_L(L^+_{t+1}), \qquad \varepsilon^L_{t+1} = \widehat{L}_{t+1} - L^+_{t+1}. \tag{55}$$

The new target reconstructed from the actual online parameters satisfies

$$\begin{aligned} \widehat{\bar{\theta}}_{t+1} &= \theta_{t+1} + \widehat{L}_{t+1} \\ &= \theta_{t+1} + (1-\tau)(\widehat{L}_t - \Delta\theta_t) + \varepsilon^L_{t+1} \\ &= \boxed{(1-\tau)\widehat{\bar{\theta}}_t + \tau\theta_{t+1} + \varepsilon^L_{t+1}.} \end{aligned} \tag{56}$$

This is the reconstructed-target identity stated in the main text. The equality holds for each realized update; it does not require the lag quantizer to be unbiased. □

Consequences. If the two networks share their initial parameters, then $\widehat{L}_0 = 0$ gives $\widehat{\bar{\theta}}_0 = \theta_0$. For comparison, define an exact EMA along the same realized online sequence by

$$\bar{\theta}^\star_{t+1} = (1-\tau)\bar{\theta}^\star_t + \tau\theta_{t+1}, \qquad \bar{\theta}^\star_0 = \theta_0. \tag{57}$$

Subtracting it from equation 56 yields, with $d_t = \widehat{\bar{\theta}}_t - \bar{\theta}^\star_t$,

$$d_{t+1} = (1-\tau)d_t + \varepsilon^L_{t+1}, \qquad d_t = \sum_{k=1}^{t}(1-\tau)^{t-k}\varepsilon^L_k. \tag{58}$$

In particular, if lag writeback is exact, the reconstructed target equals the original EMA at every step. With fixed online parameters, $\Delta\theta_t = 0$, and the unquantized lag candidate contracts as $L^+_{t+1} = (1-\tau)\widehat{L}_t$. These are algebraic properties of the relative representation; they do not assert that the actual quantized lag is error-free or necessarily more accurate than every absolute-target codec.

Why the actual online displacement is required. Suppose one instead used an intended displacement $\Delta\theta^{\text{int}}_t$ but reconstructed the target around the actual θ_{t+1}. The resulting identity would contain an additional reference-mismatch term:

$$\widehat{\bar{\theta}}_{t+1} = (1-\tau)\widehat{\bar{\theta}}_t + \tau\theta_{t+1} + (1-\tau)(\Delta\theta_t - \Delta\theta^{\text{int}}_t) + \varepsilon^L_{t+1}. \tag{59}$$

Using equation 52 removes this extra term and leaves only the explicitly defined lag quantization error.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 MAIN COMPARISON IN TABULAR FORM

Table 11 reports the final scores corresponding to the main-text learning curves, using the budgets and aggregation protocol in Appendix B.8. Each benchmark is averaged equally over tasks within a seed and then over five seeds. BRC-FP32 is the high-precision reference. The remaining baseline methods are adapted within the same BRC training pipeline with FP8 persistent storage, as in the main comparison.

Table 11: Final performance corresponding to the main-text learning curves. MT50 uses success rate; DMC-Hard and HB-Medium use normalized return. Each entry is a five-seed mean with its 95% confidence interval, written as $\mu_{[\ell,u]}$. Higher is better.

Method	MT50 500K steps/task	DMC-Hard 500K steps/task	HB-Medium 1M steps/task
BRC-FP32	$0.900_{[0.879,0.918]}$	$0.680_{[0.662,0.696]}$	$0.950_{[0.929,0.968]}$
Independent SR	$0.688_{[0.665,0.706]}$	$0.412_{[0.393,0.427]}$	$0.568_{[0.545,0.586]}$
FP16RL	$0.341_{[0.325,0.354]}$	$0.282_{[0.267,0.293]}$	$0.301_{[0.283,0.315]}$
ECO	$0.809_{[0.789,0.827]}$	$0.448_{[0.430,0.464]}$	$0.658_{[0.636,0.678]}$
FlashOptim	$0.016_{[0.004,0.027]}$	$0.012_{[0.001,0.023]}$	$0.014_{[0.001,0.027]}$
MS-AMP O2 (FP8-LM)	$0.134_{[0.113,0.154]}$	$0.102_{[0.081,0.122]}$	$0.068_{[0.045,0.091]}$
COAT	$0.113_{[0.094,0.132]}$	$0.085_{[0.066,0.104]}$	$0.139_{[0.118,0.160]}$
Correlated Quantization	$0.716_{[0.695,0.737]}$	$0.368_{[0.348,0.388]}$	$0.702_{[0.680,0.724]}$
Signed-SR$_\epsilon$	$0.780_{[0.766,0.797]}$	$0.508_{[0.495,0.523]}$	$0.612_{[0.598,0.629]}$
CoSR (ours)	$0.907_{[0.892,0.925]}$	$0.666_{[0.653,0.682]}$	$0.943_{[0.928,0.960]}$

D.2 BASELINES AT ORIGINAL PRECISION

The main experiments3 and figure4 compare baseline mechanisms under a common FP8 weight-storage setting. To complement this comparison, we evaluate FP16RL, ECO, FlashOptim, MS-AMP O2, and COAT using their original precision configurations on MT50, retaining their prescribed weight, optimizer, and auxiliary-state formats. We use the same BRC architecture, batch size, update frequency, and 500K-step per-task budget, with five random seeds. Table 12 shows that CoSR still remains closest to BRC-FP32 in success rate and achieves the best overall trade-off between learning performance, GPU memory requirements, and update latency.

Table 12: MT50 comparison with baselines at original precision (500K steps per task). Success rates are five-seed means with 95% bootstrap confidence intervals, written as $\mu_{[\ell,u]}$. State and peak memory are reported in GiB; latency is in milliseconds per update. Methods marked † retain their FP8 configurations as rounding references.

Method	MT50 Success ↑	State GiB ↓	Peak GiB ↓	Latency ms ↓
BRC-FP32	$0.900_{[0.879,0.918]}$	2.03	4.68	34.29
Independent SR†	$0.688_{[0.665,0.706]}$	0.53	1.75	20.08
FP16RL	$0.812_{[0.801,0.824]}$	1.53	4.35	32.31
ECO	$0.829_{[0.799,0.837]}$	1.66	2.88	33.23
FlashOptim	$0.216_{[0.193,0.233]}$	1.18	14.17	73.81
MS-AMP O2 (FP8-LM)	$0.564_{[0.541,0.588]}$	1.16	2.38	28.59
COAT	$0.368_{[0.340,0.383]}$	1.30	2.50	29.90
Correlated Quantization†	$0.716_{[0.695,0.737]}$	0.53	1.75	23.10
Signed-SR$_\epsilon$†	$0.780_{[0.766,0.797]}$	0.53	1.75	23.29
CoSR (ours)	$0.907_{[0.892,0.925]}$	0.53	1.74	23.77

D.3 MEMORY AND UPDATE LATENCY

Table 13 provides absolute costs for the MT50 comparison. Persistent-state memory includes weights, optimizer states, target states, scales, and any master or compensation states. Peak memory additionally includes temporary training buffers. Update latency is measured after warmup with device synchronization, excluding compilation, environment interaction, and evaluation.

Table 13: MT50 memory and update cost at matched architecture, batch size, and update frequency. Lower is better for all three metrics. Memory is reported in GiB (2^{30} bytes); latency is in milliseconds per update.

Method	Persistent-state memory (GiB)	Peak training memory (GiB)	Update latency (ms)
BRC-FP32	2.03	4.68	34.29
Independent SR	0.53	1.75	20.08
FP16RL	1.03	3.85	27.57
ECO	0.53	1.75	22.57
FlashOptim	0.66	13.65	68.85
MS-AMP O2 (FP8-LM)	0.53	1.75	22.67
COAT	0.55	1.75	22.79
Correlated Quantization	0.53	1.75	23.10
Signed-SR$_\epsilon$	0.53	1.75	23.29
CoSR (ours)	0.53	1.74	23.77

D.4 CAPACITY SCALING UNDER A MEMORY BUDGET

Table 14 records the configuration of the joint 165-task comparison. Both methods use 1M environment interactions per task. The comparison approximately matches peak JAX-allocated training memory, using the BRC-FP32$_{4096}$ run as the budget reference. Scores are reported separately for the three task groups. Main result figure is shown here5

Table 14: Joint training on 165 tasks. BRC-FP32$_{4096}$ and CoSR are compared at approximately matched peak training memory; the nominal budget is approximately 4.7 GiB. The other two columns are width-8192 references. Task-group scores follow the main-text capacity-scaling experiment.

Quantity	BRC-FP32$_{4096}$	BRC-FP32$_{8192}$	Independent SR	CoSR
Critic width	4096	8192	8192	8192
Critic parameter count	136,156,522	540,743,018	540,743,018	540,743,018
Total trainable parameter count	136,325,778	540,912,274	540,912,274	540,912,274
Environment steps per task	1M	1M	1M	1M
Total training interactions per seed	165M	165M	165M	165M
Persistent-state memory (GiB)	2.03	8.06	2.03	2.03
Peak training memory (GiB)	4.68	12.57	4.65	4.65
MT50 score (50 tasks)	$0.77_{[0.76,0.78]}$	$0.89_{[0.87,0.91]}$	$0.72_{[0.71,0.74]}$	$0.88_{[0.87,0.90]}$
DMC score (30 tasks)	$0.51_{[0.49,0.52]}$	$0.64_{[0.63,0.65]}$	$0.49_{[0.47,0.51]}$	$0.65_{[0.63,0.67]}$
ShadowHand score (85 tasks)	$0.72_{[0.71,0.73]}$	$0.84_{[0.82,0.85]}$	$0.69_{[0.67,0.70]}$	$0.83_{[0.81,0.84]}$