PRESERVING PLASTICITY IN CONTINUAL ROBOT LEARNING FROM PIXELS WITH STIEFEL-CONSTRAINED OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A robotic agent that continually learns new tasks from pixel-based inputs must maintain the plasticity of deep neural networks that process high-dimensional observations. We show that existing methods based on regularizing neural network parameters to mitigate loss of plasticity remain insufficient in the continual visual-control settings. We show that changing the optimizer from AdamW to Skewon (Solonko et al., 2026) mitigates the loss of plasticity. The Skewon optimizer constrains the parameter matrices of the deep neural network to the Stiefel manifold, meaning their columns are orthonormal. We show that this optimizer does so by preserving the input–output Jacobian norm alongside sustained learning. We demonstrate continual learning from images for up to 100 million frames for locomotion tasks using the proximal policy optimization algorithm and Skewon. We further demonstrate its efficacy in simulated manipulation tasks. We introduce a new optimizer: Gram-Skewon, which preserves pretrained weight Gram matrices, which is applied to continual behavior cloning for manipulation tasks using the SO-101 robotic arm in the physical world and outperforms AdamW. Gram-Skewon achieves 62% success versus 37% for AdamW in the real-world task chain.

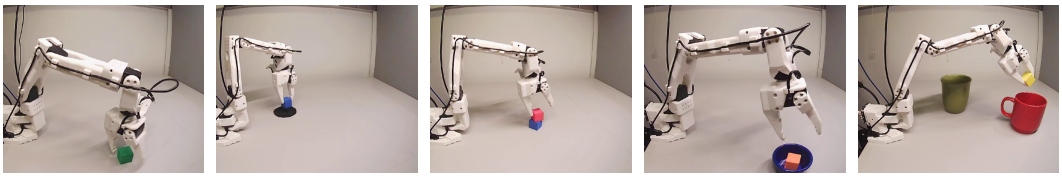
1 INTRODUCTION

An intelligent agent interacts with an ever-changing world and learns continually from it. Imagine a robotic arm deployed in a warehouse that learns a stream of manipulation tasks: picking boxes, placing them on conveyor belts or in containers, and stacking them for storage. Reinforcement learning (RL) models learning through interaction (Sutton & Barto, 1998), and agents using deep neural networks (DNNs) have been remarkably successful (Ouyang et al., 2022; Kaufmann et al., 2023). A robotic deep RL agent must complete such task streams from high-dimensional observations using its actuators over long time horizons.

The big world hypothesis says that an agent's capacity is limited relative to the complexity of its environment (Javed & Sutton, 2024). A warehouse robot must manipulate many different objects while wear and tear change its movement dynamics. Its success therefore depends on remaining able to learn as new tasks and conditions arise, making plasticity essential. We study this problem in online continual learning, where the agent learns from data obtained through current interactions with the environment.

Deep RL agents lose their ability to learn, which is called loss of plasticity (Dohare et al., 2024). This phenomenon is observed when the data distribution, from which the agent learns, changes over time. This has been well studied and documented for robotic control from low-dimensional proprioceptive states (Dohare et al., 2024; Cisse et al., 2017). This is so because the matrices that parameterize the DNNs can exhibit pathological behavior that leads to saturation of neuronal activations. However, loss of plasticity in control from high-dimensional observations, such as images used for robotic manipulation, remains under-studied. As tasks change, shifts in observation distributions and control requirements demand continued adaptation of the DNN's representations and policy, making plasticity essential.

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) Task 1: Pick (b) Task 2: Place (c) Task 3: Stack (d) Task 4: Cup (e) Task 5: Sort

Figure 1: We fine-tune SmolVLA sequentially on these tasks using our optimizer: **Gram-Skewon**. Our optimizer achieves a 62% success rate as opposed to a 37% success rate achieved by AdamW on the last task.

We demonstrate that preserving a property $W^\top W = I$ of a matrix $W \in \mathbb{R}^{m \times n}$ mitigates the loss of plasticity. Such matrices are called Stiefel matrices, and their set is denoted by $\mathrm{St}(m, n)$. A deep RL agent learns by taking steps in the direction of solving a task by updating its parameter matrices using an optimization algorithm. We show that maintaining this property while taking these steps makes the DNN well-behaved for continual learning. It ensures that there are sufficiently many directions for the matrices to change in despite the ever-changing and never-ending nature of the continual learning problem. We apply a new optimizer: Skewon (Solonko et al., 2026) that constrains the matrices of the DNN to the Stiefel manifold and mitigates the loss of plasticity.

The Skewon optimizer ensures that the DNN remains plastic even after learning from 100 million frames in robotic control environments with image observations and changing dynamics when trained using proximal policy optimization (PPO) (Lillicrap et al., 2015). Furthermore, we show that it results in continual learning in simulated robotic manipulation tasks, with pixel observations. We also provide a toy linear-quadratic-regulator experiment showing that Skewon outperforms AdamW despite hyper-parameter tuning in tasks where the optimal actions fluctuate in magnitude. This also motivates the connection between input-output Jacobian norm and plasticity. Finally, we introduce Gram-Skewon, a new optimizer that preserves pretrained weight Gram matrices $W^\top W$ to continually fine-tune vision-language-action models while maintaining plasticity. Using Gram-Skewon to continually fine-tune SmolVLA (Shukor et al., 2025), with nearly 100 million tunable parameters, for the physical SO-101 robotic arm yields significant improvements over AdamW (Figure 1).

2 Related Work

Continual learning has been an active area of research for decades (Ring, 1994; 2005; Carlson et al., 2010; Nguyen et al., 2017; Baker et al., 2023; Abel et al., 2023). Dohare et al. (2024) systematically study loss of plasticity in deep RL agents. Dormant neurons exhibit low average absolute activation relative to other neurons in the same layer over the input distribution (Sokar et al., 2023). They show that resetting the parameters associated with the neurons and bringing them back from dormancy mitigates loss of plasticity. This is a data-based approach: the neuron activations are measured for varying data input (Sokar et al., 2023; Ceron et al., 2024). A similar data-based approach uses the *effective feature rank*, which is a proxy for the rank of features of a DNN (Kumar et al., 2020). Prakash et al. (2026) propose to regularize the effective rank to maintain plasticity. Similarly, Lewandowski et al. (2025) regularize layer-wise Jacobians, which are data-dependent. Estimating data-dependent statistics can introduce additional computational overhead and sampling variance.

Parallelly, there are parameter-based interventions which do not use data samples to maintain plasticity. The simplest intervention is to regularize with the L2-norm, which prevents the parameters from approaching very high magnitudes and has the downstream effect of keeping the data-based measures such as neuron activation, effective rank and layer-wise Jacobian spectra well behaved (Lyle et al., 2024; Prakash et al., 2026). Lyle et al. (2024) demonstrate how a combination of layer-norm, reward normalization, and L2 regularization mitigates loss of plasticity. Prakash et al. (2026) also combine parameter-based L2-regularization with data-dependent effective rank regularization to demonstrate state-of-the-art performance for toy image classification tasks such as CIFAR and ImageNet. Prakash et al. (2026) also show that parameter-based methods are 55.04% faster than their method even in a toy setting. Ash & Adams (2020) also reset weights and add random noise in a method named Shrink & Perturb methodology, which is also data-dependent. Chaudhry et al. (2020) show that retracting the parameter matrices to the Stiefel manifold enables continual learning in image classification. Similarly, Chung et al. (2024) regularize the network by penalizing its

squared L2-distance, or distance under the Frobenius norm, to the Stiefel manifold: $||WW^\top - I||_F$ which they call Parseval regularization. They present empirical results in the low-dimensional RL setting. Chung et al. (2024) also show that the Parseval regularization method outperforms Layernorm, Shrink & Perturb (Ash & Adams, 2020), and regenerative regularization (Kumar et al., 2023). Recently Rosseau et al. (2026) have also shown the efficacy of Parseval-like regularization for continual RL. They achieve this by ensuring that the input-output Jacobian remains well conditioned.

Our work also introduces and evaluates a parameter-based intervention: constraining the matrices to the Stiefel manifold. Gram-Skewon preserves the pretrained weight geometry without requiring an initial projection onto the Stiefel manifold, allowing adaptation to begin from the original pretrained model. We move the intervention of constraining the parameter matrices from the loss function, via a regularizing term, to the optimizer and see significantly improved returns. Most previous work on the preservation of plasticity in RL for robotic control shows results in the setting where the agent has access to low-dimensional proprioceptive states (Dohare et al., 2024; Chung et al., 2024) and, to the best of our knowledge, we are the first to conduct experiments in such a setting. We first show that deep RL agents in environments with high-dimensional observations face a challenge maintaining plasticity under all the popular interventions (Lyle et al., 2022; Chung et al., 2024; Prakash et al., 2026). We then show the efficacy of the Skewon optimizer for pixel-based control environments (Todorov et al., 2012; McInroe & Garcin, 2025).

3 Background

We present technical background for measuring loss of plasticity and its connection to the input-output Jacobian matrix. Specifically, we define the input-output Jacobian and how it tracks plasticity. Additionally, we describe the optimizer that takes gradient steps under the constraint $W^\top W = I$.

3.1 Measuring Loss of Plasticity

We measure loss of plasticity through declining returns and study the input-output Jacobian norm at each task's start. Stiefel constraints fix the weight matrices' singular values at one, motivating our study of how this structure affects the Jacobian and plasticity. The Jacobian of a scalar function $f : \mathbb{R}^n \to \mathbb{R}$, denoted by $\nabla f(x)|_{x=v} = \partial f(x)/\partial x|_{x=v} = \nabla_x f(x)^\top \in \mathbb{R}^{1\times n}$, is a vector that captures how the function changes along different dimensions in the neighborhood of the input vector v. For a function with a vector output of m dimensions, the Jacobian is an $m \times n$ matrix. We define it for a simple feed-forward network, without biases. It has parameter matrices $W_1, W_2, \dots W_L$ and the following parameterization: $F(x, W_1, W_2, \dots W_L) = W_L\Phi(W_{L-1} \dots \Phi(W_1 x))$ where Φ is a non-linearity applied element-wise. A neuron's activation at index i for layer l is $\phi_{i,l}(x) = \Phi(W_l \dots \Phi(W_1 x))[i]$. The feature vector at layer l for data point x is the concatenation of activations $\phi_{i,l}(x)$ across all i and is denoted by $\phi_l(x)$. Define $\phi_0(x) = x$. The diagonal matrix of activation derivatives at layer l is

$$D_l(x) = \text{diag}(\Phi'(W_l\phi_{l-1}(x))), \qquad l = 1, \dots, L-1,$$

where Φ' is applied element-wise to the pre-activation vector $W_l\phi_{l-1}(x)$. Thus, the Jacobian is defined as: $\partial F(x; W_1, \dots, W_L)/\partial x = W_L D_{L-1}(x) W_{L-1} \cdots D_1(x) W_1$.

Xiao et al. (2018) condition the Jacobian singular values to improve image classification accuracy using DNNs. Rosseau et al. (2026) apply this to continual learning to provide further theoretical analysis. Intuitively, a larger norm of the input-output Jacobian matrix for a DNN means that the function output changes significantly with perturbations of its input and therefore the agent can use rewards from the environment along with these perturbations to learn. In contrast, a DNN whose outputs do not change much across perturbations cannot obtain this reward-based feedback. Further, restricting the parameter matrices to the Stiefel manifold fixes their singular values at unity, preventing the linear transformations from introducing ill-conditioning and potentially improving the conditioning of the input-output Jacobian, although its spectrum also depends on the activation derivatives $D_l(x)$.

3.2 Gradient Updates with Retraction to Stiefel Manifold

Our parameter-based approach to mitigating loss of plasticity introduces two key perspectives: treating matrices as operators rather than vectors and imposing a strict constraint rather than regularizing the matrices. At their core, deep neural networks are a composition of matrix multiplications interposed by non-linearities. A deep RL agent learns by updating its parameters, the weights of the matrices, by taking gradient ascent steps in the direction of its objective. The Muon optimizer (Jordan et al., 2024), which treats these matrices as operators rather than a flattened vector, has shown improved empirical performance over AdamW in supervised learning settings. Another relevant extension of this approach is to restrict the matrices to the *Stiefel manifold* (Absil et al., 2008; Wen & Yin, 2013; Bernstein, 2025a), the set of matrices in $\mathbb{R}^{d\times k}$ with orthonormal columns:

$$\mathrm{St}(d,k) := X \in \mathbb{R}^{d\times k} : X^\top X = I_k, \qquad d \geq k.$$

A standard Euclidean gradient update generally does not preserve this constraint. Optimization on the Stiefel manifold therefore constructs an update direction in the tangent space,

$$T_X \,\mathrm{St}(d,k) = Z \in \mathbb{R}^{d\times k} : X^\top Z + Z^\top X = 0,$$

and applies a retraction to map the resulting step back onto the manifold (Absil et al., 2008).

Let $X \in \mathbb{R}^{d\times d}$ be a square parameter matrix constrained to the Stiefel manifold, and let D_X denote the Euclidean gradient of the objective being maximized. One such update uses the Cayley retraction:

$$\Omega_X := D_X X^\top - X D_X^\top, \qquad X^+ = \left(I_d - \frac{\eta}{2}\Omega_X\right)^{-1}\left(I_d + \frac{\eta}{2}\Omega_X\right) X, \tag{1}$$

where η is the learning rate, Ω_X is skew-symmetric, and $\Omega_X X$ is a tangent ascent direction. The superscript $+$ denotes the updated parameter matrix after one step. This update preserves orthogonality, ensuring that the constrained parameter matrices remain on the Stiefel manifold throughout training, provided they are initialized on the manifold (Absil et al., 2008; Wen & Yin, 2013). This line of work has recently been applied to optimization of large language models (Bernstein, 2025b;a; Thinking Machines Lab, 2026).

In practice, directly evaluating the Cayley update requires a matrix equation to be solved with $O(d^3)$ cost, which can be expensive for large parameter matrices of the DNNs. For our empirical analysis, we use **Skewon** (Solonko et al., 2026), which provides a closed-form solution for the spectral-norm-constrained tangent update underlying Stiefel Muon. Given a gradient or momentum matrix M_X oriented toward ascent, its update takes the form

$$N_X := \frac{1}{2}\left(M_X X^\top - X M_X^\top\right), \qquad \Xi_{\mathrm{opt}} = \mathrm{polar}_0(N_X)X,$$
$$X^+ = R_X\left(\eta \Xi_{\mathrm{opt}}\right),$$

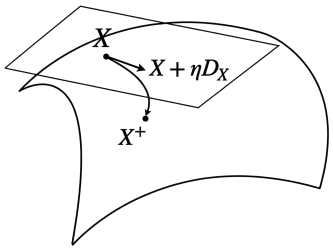


Figure 2: Vanilla gradient steps can move the matrix off the manifold; the retraction pulls it back.

where $\mathrm{polar}_0(N_X) = UV^\top$ for the compact SVD $N_X = U\Sigma V^\top$, with zero mapped to zero. Here, R_X denotes a retraction. Skewon eliminates the iterative optimization previously used to determine the tangent direction; its practical implementation evaluates the polar factor using Newton-Schulz iterations (Solonko et al., 2026). Using a QR or polar retraction also avoids the explicit Cayley solve, although this does not remove all cubic-cost matrix operations for square matrices. While our theoretical analysis uses the Cayley retraction for its concise mathematical form, our empirical analysis uses Skewon with retraction. We combine these insights, having structured parameter matrices and an optimizer effective in gradient-based learning, with the underlying low-dimensional structure in RL problems with high-dimensional observations.

4 Environment Design for Continual Reinforcement Learning

We establish how AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2017) can lead to loss of plasticity for control from high-dimensional observations. We first introduce a general multi-task reinforcement learning setting. Each task, indexed by τ, is defined by the tuple

$$\mathcal{M}_\tau = \langle \mathbb{R}^{d_s}, \mathbb{R}^{d_a}, (f_s^\tau, \Sigma_\tau), (\mathbb{R}^{d_o}, f_o^\tau, \epsilon), r_\tau, \gamma, s_0 \rangle,$$

where $\mathbb{R}^{d_s}$, $\mathbb{R}^{d_a}$, and $\mathbb{R}^{d_o}$ are the shared Euclidean state, action, and observation spaces, respectively, $\gamma \in [0, 1)$ is the discount factor, and $s_0 \in \mathbb{R}^{d_s}$ is the start state. The state transition and observation functions, $f_s^\tau : \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to \mathbb{R}^{d_s}$, $f_o^\tau : \mathbb{R}^{d_s} \to \mathbb{R}^{d_o}$, may be nonlinear and determine the dynamics and observations through

$$s_{t+1} = f_s^\tau(s_t, a_t) + \Sigma_\tau X_t, \;\; X_t \sim \mathcal{N}(0, I_{d_s}), \qquad o_t = f_o^\tau(s_t) + \epsilon Y_t, \;\; Y_t \sim \mathcal{N}(0, I_{d_o}).$$

where $\Sigma_\tau \in \mathbb{R}^{d_s \times d_s}$ and $\epsilon \geq 0$ determine the process and observation noise, respectively, and the noise variables are independent across time and of one another. The reward function $r_\tau : \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to \mathbb{R}$ specifies the reward for taking action a_t in state s_t.

This setting models control from high-dimensional observations, with $d_o \gg d_s$. For example, the state of a humanoid robot walking forward can be described by its joint positions, joint velocities, and location, whereas an image of the robot contains thousands or tens of thousands of pixels (Figure 3). The first description is the underlying state, while the second is the observation available to the agent. We consider policies that select actions from the current observation, $a_t \sim \pi(\cdot \mid o_t)$; a deterministic policy is a map $\pi : \mathbb{R}^{d_o} \to \mathbb{R}^{d_a}$.

For each task, the agent seeks a policy that maximizes the expected cumulative discounted reward:

$$J_\tau(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_\tau(S_t, A_t) \,\middle|\, S_0 = s_0\right],$$

where the expectation is over trajectories generated by π in $\mathcal{M}_\tau$. The corresponding state-value function is $V_\tau^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_\tau(S_t, A_t) \,\middle|\, S_0 = s\right]$. In our deep RL setting, the agent uses the on-policy PPO algorithm (Schulman et al., 2017), with a DNN policy and a DNN critic that estimates discounted returns from the agent's observations. In the continual learning setup, the agent encounters the tasks $\mathcal{M}_1, \mathcal{M}_2, \ldots$ sequentially and carries its learned parameters across task changes. We refer to a sequence of tasks as a chain. Tasks may differ in their transition dynamics, observation maps, noise, or rewards, while sharing the same state, action, and observation spaces. This construction follows the rich-observation control perspective, in which low-dimensional latent dynamics are accessed through high-dimensional nonlinear observations (Mhammedi et al., 2020).

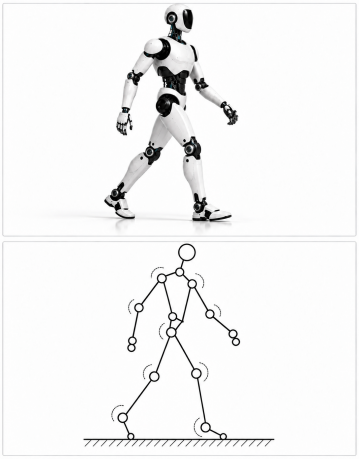


Figure 3: Suppose the agent's goal is to control a humanoid to walk. Top: a picture of the humanoid robot (high-dimensional). Bottom: a schematic of its pose denoting the underlying structure (low-dimensional) [1].

4.1 AdamW Causes Loss of Plasticity

We specialize the above multi-task setting to LQR tasks (Anderson & Moore, 2007; Fazel et al., 2018) by choosing linear state transitions and negative quadratic rewards:

$$f_s^\tau(s, a) = G_\tau s + H_\tau a, \qquad r_\tau(s, a) = -\left(s^\top Q_\tau s + a^\top R_\tau a\right),$$

where $G_\tau \in \mathbb{R}^{d_s \times d_s}$, $H_\tau \in \mathbb{R}^{d_s \times d_a}$, $Q_\tau \in \mathbb{R}^{d_s \times d_s}$ is symmetric positive semidefinite, and $R_\tau \in \mathbb{R}^{d_a \times d_a}$ is symmetric positive definite. The observation map f_o^τ remains potentially nonlinear, allowing the agent to control the underlying LQR system from images.

To compare AdamW and Skewon in this simplified continual learning setting, we use an environment with eight tiles, each containing a blob that the agent must bring to the target marked by an X (Figure 4). The two-dimensional displacements of the eight blobs from their targets form a 16-dimensional state, while the action space is 8-dimensional, so a single action component can influence multiple blobs. The matrices G_τ and H_τ determine how the blob positions and actions influence subsequent movement, analogous to strings coupling the blobs. The agent receives a flattened image of this state with observation dimension $d_o = 2^{13}$. For each task τ, we randomly sample the matrices determining the underlying dynamics while ensuring that the LQR problem remains solvable.

[1] Image generated by ChatGPT.

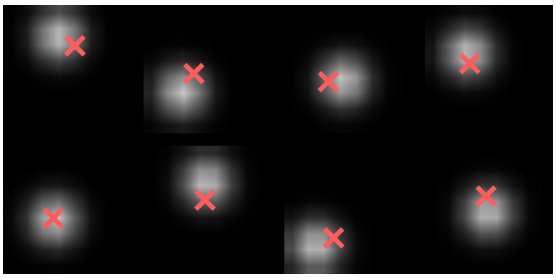


Figure 4: The agent has to move the 8 light blobs to perfectly and simultaneously align with their associated red cross marks with correlated 8-dimensional actions. We vary how the agent is rewarded and how the correlated actions affect the state (H_τ) at each step in the chain.

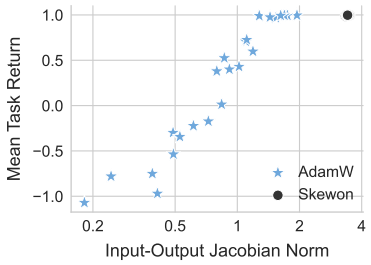


Figure 5: Mean return against the mean input–output Jacobian norm for 24 chains. We see that AdamW has lower performance on average while Skewon (all 24 overlapping) performs near optimally.

Choosing a simple LQR testbed allows us to simulate in JAX (Bradbury et al., 2018; McInroe & Garcin, 2025) and present comprehensive hyper-parameter sweeps to compare AdamW and Skewon optimizers. We train Skewon and AdamW agents across 24 chains of 21 tasks, selecting the best hyper-parameter configuration for each optimizer and chain from 12 settings evaluated across 3 seeds. Figure 5 plots mean return against the input-output Jacobian norm (Section 3.1): Skewon maintains plasticity and a large Jacobian norm across all chains, while AdamW exhibits a log-linear relationship between the norm and mean task return.

5 Skewon Mitigates Loss of Plasticity

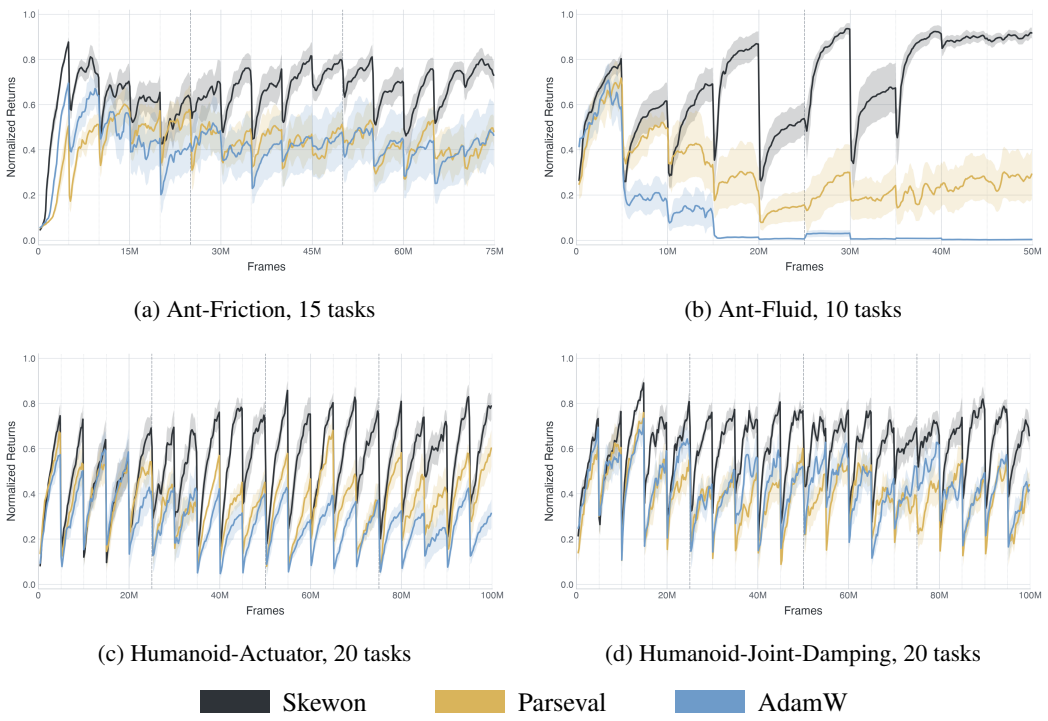
Despite hyper-parameter tuning and the use of layer normalization, reward normalization, L2 regularization, and Swish activations, AdamW still loses plasticity. Skewon sustains learning and outperforms both AdamW and Parseval across all four continual locomotion variants (Figure 6), demonstrating gains beyond the standard plasticity-preserving interventions evaluated here. We present two examples of deep RL agents learning in simulation: locomotion tasks in MuJoCo environments (Todorov et al., 2012) and manipulation tasks using robotic arms simulated using the ManiSkill framework (Tao et al., 2024; Xiang et al., 2020). In both cases, the agent observes pixels, except that in the ManiSkill example it also has access to the robotic arm's pose, but it does not have access to privileged information such as the location of objects. In both these scenarios the Skewon optimizer outperforms AdamW.

5.1 MuJoCo Locomotion Tasks

We compare Skewon with AdamW (Loshchilov & Hutter, 2017) and Parseval (Chung et al., 2024) across four continual locomotion variants with changing dynamics.

1. **Ant-Friction:** task sequences for the ant environment with 15 tasks with varying friction coefficients which govern the contact forces between the ant legs and the floor. The friction coefficients are sampled from a bimodal distribution (similar to (Dohare et al., 2024)).
2. **Ant-Fluid:** task sequences for the ant environment with 10 tasks in which the ant is immersed in a fluid, with density sampled uniformly from $[0.5, 1.0]$ while the friction coefficient continues to vary bimodally as above.
3. **Humanoid-Actuator:** task sequences of the humanoid environment with 20 tasks with varying actuator gear scales, which multiply the torque each motor delivers for a given control signal. The scale is sampled from a bimodal distribution with bands $[0.2, 0.3] \cup [2.4, 2.5]$.
4. **Humanoid-Joint-Damping:** task sequences of the humanoid environment with 20 tasks with varying joint damping coefficients, which govern how strongly rotation about each joint is resisted. The coefficient is sampled from a bimodal distribution with bands $[2, 8] \cup [40, 80]$.

We train on ten sampled task sequences per variant and report mean returns across sequences. Each task lasts 5 million frames, and the agent is trained using PPO. All methods use a CNN-based en-



(a) Ant-Friction, 15 tasks

(b) Ant-Fluid, 10 tasks

(c) Humanoid-Actuator, 20 tasks

(d) Humanoid-Joint-Damping, 20 tasks

Figure 6: Mean normalized return across ten task sequences for each continual locomotion variant. Within each task, returns are rescaled to $[0, 1]$ using the pooled minimum and maximum attained by the three methods, measuring relative performance on that task. Training is stopped after an observed separation between Skewon and the baselines.

coder with separate actor and critic heads. We provide further implementation and hyper-parameter details in Section B. The CNN encoder ends with a feedforward layer followed by a tanh activation. Training uses the standard PPO objective, including entropy regularization, together with the evaluated regularizers and no additional auxiliary objectives.

Hyper-parameters for Skewon, Parseval, and AdamW are tuned separately for Ant and Humanoid. For AdamW, we tune weight decay, the entropy coefficient, the number of mini-batches, and the learning rate. We also vary the encoder width. We apply the recommended interventions for preserving plasticity from past work (Dohare et al., 2024; Lyle et al., 2024; Prakash et al., 2026): layer normalization, L2 regularization, reward normalization, and Swish activation (Ramachandran et al., 2017; Elfwing et al., 2018). Following these recommendations, AdamW and Skewon share the architecture, layer normalization, reward normalization, and activation function.

Skewon's advantage persists despite tuning AdamW and applying the listed plasticity interventions (Figure 6). Across all four variants, Skewon pulls ahead of both baselines early in the chain and sustains its relative performance as tasks accumulate. The separation is clearest on Ant-Fluid: AdamW's normalized return declines to near the pooled minimum (0.010 ± 0.003 by the end of the chain), while Skewon maintains higher returns than both AdamW and Parseval. On both humanoid variants, Skewon maintains its advantage at 95–100M frames after 19 dynamics changes, whereas AdamW and Parseval plateau or decline in normalized return. This sustained performance is accompanied by stable Jacobian norms for Skewon and declining norms for AdamW and Parseval in the environments shown in Figure 7, supporting the connection between Jacobian behavior and plasticity studied in prior work (Hu et al., 2020; Chung et al., 2024; Rosseau et al., 2026).

5.2 ManiSkill

We evaluate Skewon and AdamW on continual manipulation with the SO-100 and Franka Panda arms, trained using PPO in ManiSkill (Tao et al., 2024).

Task	Chain		Scratch	
	AdamW	Skewon	AdamW	Skewon
Grasp	$\mathbf{.70}_{\pm.05}$	$.64_{\pm.05}$	**.88**	.75
Place	$.75_{\pm.08}$	$\mathbf{.92}_{\pm.04}$	.88	**.94**
Stack	$.31_{\pm.11}$	$\mathbf{.47}_{\pm.07}$	.56	**.69**
Cup-Align	$.03_{\pm.02}$	$\mathbf{.92}_{\pm.04}$	.81	.81
Sort	$.23_{\pm.14}$	$\mathbf{.63}_{\pm.15}$	.88	**1.00**

(a) SO-100, five tasks.

Task	Chain		Scratch	
	AdamW	Skewon	AdamW	Skewon
Grasp	$\mathbf{.92}_{\pm.04}$	$.73_{\pm.04}$	1.00	1.00
Place	$\mathbf{1.00}_{\pm.00}$	$.96_{\pm.04}$	1.00	1.00
Stack	$.75_{\pm.06}$	$\mathbf{.96}_{\pm.02}$	**.81**	.69
Pyramid	$.58_{\pm.08}$	$\mathbf{.94}_{\pm.04}$	.00	.00

(b) Franka Panda, four tasks.

Table 1: Success rates on two manipulation task chains. *Chain*: each task starts from the preceding task's final checkpoint (mean $\pm$ standard error over seeds). *Scratch*: each task is trained independently from random initialisation (single seed). Bold indicates the higher observed success rate within each comparison.

The SO-100 arm learns five tasks sequentially: **Grasp**, establishing a stable grasp on a cube; **Place**, carrying and releasing it at a sampled table target; **Stack**, placing it on another cube; **Cup-Align**, depositing it inside a walled bin; and **Sort**, placing a cube or ball in its shape-assigned bin. Sort requires identifying the held object from camera images, as its identity is excluded from the proprioceptive state. The Franka Panda arm learns four tasks sequentially: **Grasp**, grasping and lifting a cube from the table; **Place**, carrying it to a sampled goal above the table and holding it there; **Stack**, releasing it onto another cube; and **Pyramid**, balancing it atop two adjacent base cubes. For both arms, only Grasp begins with an empty gripper; subsequent tasks begin with the object held. AdamW initially performs better on Grasp for both arms, but Skewon achieves higher success on the final task of each chain: 63% versus 23% on SO-100 Sort and 94% versus 58% on Panda Pyramid. This reversal highlights Skewon's advantage in sustaining learning as the agent progresses through successive tasks. Further task details are in Section C.1.

Each task starts from the preceding task's final checkpoint, and Table 1 compares this sequential training with independent training from scratch. Skewon achieves higher success than AdamW on every task from Stack onward in both chains. On SO-100 Cup-Align, both optimizers achieve 0.81 from scratch, but AdamW reaches only 0.03 in the chain compared with Skewon's 0.92, supporting better preservation of adaptability under sequential training. Panda Pyramid provides complementary evidence of beneficial transfer: both scratch runs score zero, whereas sequential training achieves 0.94 with Skewon and 0.58 with AdamW. All training details can be found in Section C.

6 Gram-Preserving Skewon for Continual VLA Fine-Tuning

We next demonstrate that the benefits of constraining parameter matrices extend from simulated reinforcement learning to continual fine-tuning of pretrained vision-language-action (VLA) models on a physical robot. VLAs use images and natural language instructions for robotic control (Kim et al., 2024; Intelligence et al., 2025; Shukor et al., 2025; NVIDIA et al., 2025) and are commonly fine-tuned for specific tasks and embodiments (Kim et al., 2025). Pretrained weights need not satisfy $W^\top W = I$, so imposing orthonormality would alter the learned model. We instead introduce **Gram-Skewon**, which preserves each weight matrix's initial Gram matrix $G_0 = W^\top W$ and hence its singular values during adaptation. We demonstrate its effectiveness by continually fine-tuning SmolVLA (Shukor et al., 2025) on a chain of five physical manipulation tasks.

6.1 Gram-Preserving Skewon

Let $W \in \mathbb{R}^{m\times n}$ with $m \geq n$ denote one trainable weight matrix of the pretrained model, written in tall form (wide matrices are transposed, as in our Stiefel implementation). Write $\mathrm{skew}(X) = \frac{1}{2}(X - X^\top)$, let $\mathrm{Skew}(m)$ be the skew-symmetric $m \times m$ matrices, and let $\mathrm{msign}(X) = UV^\top$ for the SVD $X = U\Sigma V^\top$ be the matrix sign (orthogonal polar factor), computed by a Polar-Express iteration (Amsel et al., 2026) so that no inverse or eigendecomposition is ever formed. We extend the Stiefel constraint used by Skewon (Solonko et al., 2026) to the Gram-preserving manifold:

$$\mathcal{M}_{G_0} \;=\; \big\{\, W \in \mathbb{R}^{m\times n} \;:\; W^\top W = G_0 \,\big\}, \qquad G_0 \succ 0 \;\; (n \times n), \tag{2}$$

Table 2: Real-world success rates across the five tasks. Each policy is the final checkpoint of its five-stage continual-learning chain. Best success rate per task in **bold**.

	AdamW			Gram–Skewon		
Task	Trials	Success	Rate	Trials	Success	Rate
Pick	50	23	46.0%	50	25	**50.0%**
Place	40	39	**97.5%**	40	36	90.0%
Stack	50	6	12.0%	40	15	**37.5%**
Cup Align	40	32	80.0%	40	38	**95.0%**
Sort	60	22	36.7%	60	37	**61.7%**
Overall	240	122	50.8%	230	151	**65.7%**

with G_0 derived once from the pretrained weight. Stiefel is the special case $G_0 = I$. Therefore, given the Euclidean gradient direction $\mathcal{G}$, our goal is to solve the following problem:

$$Y^\star = \underset{Y \in \mathrm{Skew}(m),\ \|Y\|_2 \le 1}{\arg\min} \langle \mathcal{G}, YW \rangle = -\,\mathrm{msign}\big(\mathrm{skew}(\mathcal{G}W^\top)\big), \qquad \Delta W = \eta\, Y^\star W. \quad (3)$$

To this effect we introduce Algorithm 1, which is similar to Algorithm 1 by Solonko et al. (2026) for the Stiefel constraint. The derivation is provided in Section D.

6.2 Real-World Experiments

We continually fine-tune SmolVLA, with nearly 100 million tunable parameters, on five manipulation tasks using the physical SO-101 arm. Unlike the preceding RL experiments, adaptation uses demonstrations collected with a teleoperator leader arm through behavior cloning. The tasks are: **Pick**, grasping and lifting a cube; **Place**, releasing it on a black target disc; **Stack**, placing it on a blue cube; **Cup**, depositing it inside a blue cup; and **Sort**, placing a specified coloured cube into a red or green cup according to the language instruction. Only Pick begins without a grasped cube. Sort introduces six combinations of cube colour and destination, requiring language to disambiguate the goal for the same visual scene. These tasks are illustrated in Figure 1.

Across both methods, the only difference is the optimizer, and both are trained using behavior cloning (Shukor et al., 2025; Cadene et al., 2026). Behavior cloning trains on teleoperated demonstrations pairing camera images, joint states, and language instructions with demonstrated actions. SmolVLA predicts chunks of 50 actions, and fine-tuning minimises the distance to demonstrated chunks without rewards. Gram-Skewon outperforms AdamW on Stack, Cup Align, and Sort, with final-task success increasing from 36.7% to 61.7% (Table 2), extending the empirical benefits of matrix-constrained optimization to pretrained VLA adaptation on physical hardware. Gram-Skewon takes $2.03\times$ as much training time per task on average.

7 Discussion

Our results show that AdamW can lose plasticity in continual control from pixels despite the normalization and regularization interventions recommended in prior work (Dohare et al., 2024; Lyle et al., 2024; Chung et al., 2024). Replacing AdamW with Skewon (Solonko et al., 2026) improves sustained learning without introducing additional auxiliary losses or regularization coefficients. Skewon sustains learning across changing locomotion dynamics for up to 100 million frames and achieves higher success on later manipulation tasks, including 92% versus 3% for AdamW on SO-100 Cup-Align. On the physical SO-101 arm, Gram-Skewon improves SmolVLA's final-task success from 36.7% to 61.7%, extending these benefits to continual adaptation of a pretrained VLA. Building on Stiefel-constrained continual learning (Chaudhry et al., 2020), we further extend this principle to pretrained VLA adaptation through Gram-Skewon and demonstrate its benefits on physical manipulation tasks. Our Jacobian measurements provide further empirical support for the connection between network sensitivity and plasticity studied in prior work (Lewandowski et al., 2025; Rosseau et al., 2026), complementing analyses of spectral collapse (Prakash et al., 2026). These observations motivate a theoretical account of how orthogonality and Gram preservation influence the input-output Jacobian and subsequent learning.

AI USE STATEMENT

We used large language models (LLMs) as assistive tools in four ways. First, we used them to help write and debug parts of the code for our experiments. Second, we used them to improve the grammar, clarity, and readability of the manuscript. Third, we used them to check our mathematical proofs for errors. Fourth, we used them to poulate the hyper-parameters we swept over. All research ideas, the methodology, the experimental design, the analysis, and the conclusions are entirely the authors' own. Every piece of LLM-assisted content, including code, text, and proof feedback, was reviewed and verified by the authors. The authors take full responsibility for the content of this paper.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work does not involve human subjects, and no personally identifiable or sensitive information was collected or used. All datasets used are publicly available and were used in accordance with their respective licenses and terms of use. We are not aware of any direct harmful applications of this work. As with most machine learning methods, however, it could be misused or could reflect biases present in the underlying data, and we encourage responsible use and further evaluation before deployment in high-stakes settings. The authors declare no conflicts of interest or sponsorship that could have influenced the results of this work.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure our results are reproducible. The proposed method is described in Sections of the paper, and the full experimental setup, including model architectures, hyperparameters, training procedures, and evaluation protocols. Complete proofs and the assumptions behind our theoretical results appear in Appendix D. Anonymized source code, with scripts to reproduce the main experiments, is included in the supplementary materials and will be released publicly upon publication.

REFERENCES

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado Van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:50377–50407, 2023.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. In *The Fourteenth International Conference on Learning Representations*, 2026.

Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

Megan M Baker, Alexander New, Mario Aguilar-Simon, Ziad Al-Halah, Sébastien MR Arnold, Ese Ben-Iwhiwhu, Andrew P Brna, Ethan Brooks, Ryan C Brown, Zachary Daniels, et al. A domain-agnostic approach for characterization of lifelong learning systems. *Neural Networks*, 160:274–296, 2023.

Jeremy Bernstein. Modular manifolds. *Thinking Machines Lab: Connectionism*, 2025a. doi: 10.64434/tml.20250926.

Jeremy Bernstein. The modula docs, 2025b.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, 4 edition, 2017.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, Mustafa Shukor, Jess Moss, Alexander Soare, Dana Aubakirova, Quentin Lhoest, Quentin Gallouédec, and Thomas Wolf. Lerobot: An open-source library for end-to-end robot learning. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.22818.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, number 1, pp. 1306–1313, 2010.

Johan Samir Obando Ceron, Aaron Courville, and Pablo Samuel Castro. In value-based deep reinforcement learning, a pruned network is a good network. In *Forty-first International Conference on Machine Learning*, 2024.

Arslan Chaudhry, Naeemullah Khan, Puneet K. Dokania, and Philip H. S. Torr. Continual learning in low-rank orthogonal subspaces. In *Advances in Neural Information Processing Systems*, 2020.

Wesley Chung, Lynn Cherif, David Meger, and Doina Precup. Parseval regularization for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 37:127937–127967, 2024.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 854–863. PMLR, 06–11 Aug 2017.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. arXiv:1702.03118.

Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International conference on machine learning*, pp. 1467–1476. PMLR, 2018.

Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. *arXiv preprint arXiv:2001.05992*, 2020.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. Pi-0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Khurram Javed and Richard S Sutton. The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4, 2024.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model, 2024. *URL https://arxiv. org/abs/2406.09246*, 1(2):4, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In *CoLLAs*, 2023.

Alex Lewandowski, Michał Bortkiewicz, Saurabh Kumar, Andras Gyorgy, Dale Schuurmans, Mateusz Ostaszewski, and Marlos C Machado. Learning continually by spectral regularization. In *International Conference on Learning Representations*, volume 2025, pp. 34504–34530, 2025.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

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

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning, 2022.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado Van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762*, 2024.

Trevor McInroe and Samuel Garcin. Pixelbrax: Learning continuous control from pixels end-to-end on the gpu. *arXiv preprint arXiv:2502.00021*, 2025.

Zakaria Mhammedi, Dylan J. Foster, Max Simchowitz, Dipendra Misra, Wen Sun, Akshay Krishnamurthy, Alexander Rakhlin, and John Langford. Learning the linear quadratic regulator from nonlinear observations. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL `https://papers.nips.cc/paper/2020/hash/a70145bf8b173e4496b554ce57969e24-Abstract.html`.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Arjun Prakash, Naicheng He, Kaicheng Guo, Saket Tiwari, Ruo Yu Tao, Tyrone Serapio, Amy Greenwald, and George Konidaris. Spectral collapse drives loss of plasticity in deep continual learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Mark B Ring. Toward a formal framework for continual learning. In *NIPS workshop on Inductive Transfer, Whistler, Canada*, 2005.

Mark Bishop Ring. *Continual learning in reinforcement environments*. The University of Texas at Austin, 1994.

Andries Rosseau, Robert Müller, and Ann Nowé. Preserving plasticity in continual learning via dynamical isometry. *arXiv preprint arXiv:2606.09762*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 32145–32168. PMLR, 2023.

Mikhail Solonko, Molozhavenko Alexander, and Maxim Rakhuba. Muon on the stiefel manifold admits an exact closed-form update. *arXiv preprint arXiv:2608.06218*, 2026.

Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Thinking Machines Lab. Inkling: Our open-weights model, July 2026. URL `https://thinkingmachines.ai/news/introducing-inkling/`.

Saket Tiwari and George Konidaris. Effects of data geometry in early deep learning. *Advances in Neural Information Processing Systems*, 35:30099–30113, 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1):397–434, 2013.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5393–5402. PMLR, 10–15 Jul 2018.

Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin Haeffele, and Yi Ma. White-box transformers via sparse rate reduction. *Advances in Neural Information Processing Systems*, 36:9422–9457, 2023.

A Experimental Details of LQR Experiments

This appendix specifies how the LQR tasks of Section 3.1 are constructed, how they vary along a chain, and the training protocol behind Figure 5. Throughout, $d_s = 16$, $d_a = 8$, $d_o = 2^{13}$, $\gamma = 0.99$, and every episode lasts 100 steps.

A.1 Dynamics

Each task uses the linear transition $f_s^\tau(s,a) = G_\tau s + H_\tau a$ with process noise $\Sigma_\tau X_t$. The transition matrix is the discretization $G_\tau = \exp(A_\tau\,\delta)$, $\delta = 0.1$, of a continuous-time drift $A_\tau = V_\tau \text{Block Diagonal}(B_1, \ldots, B_{d_s/2}) V_\tau^\top$, where V_τ is a random orthogonal matrix and each 2×2 block $B_i = \left(\begin{smallmatrix} \sigma_i & \omega_i \\ -\omega_i & \sigma_i \end{smallmatrix}\right)$ has eigenvalues $\sigma_i \pm \mathrm{i}\,\omega_i$ with $\omega_i \sim \mathcal{U}[0.5, 5]$. Since A_τ is a normal matrix, its spectrum is exactly the set of block eigenvalues, so stability is controlled directly by the decay rates σ_i. The actuation matrix H_τ is a random point on $\mathrm{St}(d_s, d_a)$, so the eight action components push along eight orthonormal state directions, each of which moves several blobs at once. Therefore, we refer to this as the blobs being attached by "strings". The process noise is $\Sigma_\tau = 0.1\,U\,\mathrm{diag}(\lambda)U^\top$ with U random orthogonal and eigenvalues λ_i log-uniform on $[1/4, 4]$; the observation noise is $\epsilon = 0$. The state is confined to the ball $\|s - s_\tau^\star\| \le 10$ about the task goal $s_\tau^\star$ defined below, by rescaling any state that leaves it, so the cost of an uncontrolled unstable mode remains bounded.

A.2 Rewards and Goals

Each task has a goal state $s_\tau^\star$ and a feed-forward action v_τ, and the reward is the negative quadratic cost in the error coordinates,

$$r_\tau(s,a) = -\Big((s - s_\tau^\star)^\top Q_\tau (s - s_\tau^\star) + (a - v_\tau)^\top R_\tau (a - v_\tau)\Big),$$

with $Q_\tau = U\,\mathrm{diag}(q)U^\top$ and $R_\tau = 0.1\,\rho_\tau U'\,\mathrm{diag}(\varrho)U'^\top$ for random orthogonal U, U', eigenvalues q_i, ϱ_i log-uniform on $[1/4, 4]$, and a task-type dependent effort-price multiplier ρ_τ. The goal is always reachable: given v_τ we set $s_\tau^\star = (I - G_\tau)^{-1} H_\tau v_\tau$, the state that the constant action v_τ holds in equilibrium, so that $(I - G_\tau)s_\tau^\star = H_\tau v_\tau$ exactly. In error coordinates the problem is the original LQR problem, so the optimal policy is $a = v_\tau - K_\tau(s - s_\tau^\star)$ with K_τ the gain of the discounted discrete-time algebraic Riccati equation for $(G_\tau, H_\tau, Q_\tau, R_\tau)$, which we use as the reference in all normalized returns (Bertsekas, 2017). Every episode starts from $s_0 \sim \mathcal{N}(s_\tau^\star, I_{d_s})$, drawn independently for each of the 512 parallel environments.

Despite being a fairly complex definition of an environment it has two benefits: **1** it is solvable analytically and we can obtain the optimum returns analytically as well, **2)** the dynamics are stable meaning that the state does not appraoch very large values in finite time. This is so because it is a long studied model in control theory (Anderson & Moore, 2007).

A.3 How Tasks Vary Along a Chain

A chain alternates two task types, $\mathcal{M}_1, \mathcal{M}_2, \ldots = \mathrm{F}, \mathrm{H}, \mathrm{F}, \mathrm{H}, \ldots$, chosen so that the optimal input-output gain of the policy is driven down and back up at every hop. This also mimics a bimodal distribution of task parameters (Dohare et al., 2024). **H-tasks** ("must control") have one unstable block, $\sigma_i = +0.3$ for a randomly chosen i (doubling time of about 23 steps) and $\sigma_j \sim -\mathcal{U}[0.05, 1.5]$ for the others, effort price $\rho_\tau = 0.1$, and $v_\tau = 0$, so $s_\tau^\star = 0$. Feedback is essential: the optimal cost is about 2% of that of a constant policy, $\|K_\tau\| \approx 3.1$, and the closed-loop spectral radius is 0.82–0.92. **F-tasks** ("fluid") have a heavily damped plant, $\sigma_j \sim -\mathcal{U}[20, 30]$ for every block (spectral radius 0.12), nominal effort price $\rho_\tau = 1$, and a feed-forward action $v_\tau = |v|\,u_\tau$ with u_τ a random unit vector. Because the state decays by itself, feedback buys almost nothing ($\|K_\tau\| \approx 0.2$) while holding the goal against the damping requires a constant output of magnitude $|v|$: a task of low input

sensitivity and large output magnitude, the regime of a legged robot pushing through drag. The H-task that follows is the opposite regime, and a policy that arrives still emitting the F-task's constant drives the unstable plant into the boundary of the state ball, where the return no longer depends on the action. Within a chain the H-plant and the F-plant are each drawn once and reused, so the only quantity that changes from one F-task to the next is the push v_τ; this isolates the magnitude $|v|$ as the single knob that sets the difficulty of the hop.

We vary $|v|$ in two ways. In the *fixed* chains every F-task uses the same magnitude, $|v| \in \{2.75, 3, 3.25, 3.5, 3.75, 4\}$. In the *random* chains the magnitude of each F-task is drawn independently as $|v| \sim \mathcal{U}[0.5\,\bar{v}, 1.5\,\bar{v}]$ around a nominal $\bar{v} \in \{1.25, 1.5, \ldots, 4\}$ with spacing 0.25, refined to spacing 0.125 over $[2.125, 3.375]$ where AdamW transitions from recovering to trapped. **This gives** $6 + 18 = 24$ **chains per optimizer**, and 3 seeds per chain; Figure 5 shows one point per chain averaged over 3 seeds.

A.4 Observations

The observation map f_o^τ renders the state as a flat vector of 2^{13} pixels. The 16 state coordinates are paired into 8 tiles of 8×8 pixels; tile j contains a Gaussian blob of width 0.3 (in tile coordinates $[-1, 1]^2$) centred at $\big(0.7\tanh((s - s_\tau^\star)_{2j}/2),\ 0.7\tanh((s - s_\tau^\star)_{2j+1}/2)\big)$, so the goal is the centre of every tile and the $\tanh$ keeps the blob inside the tile at the boundary of the state ball. The 512 foreground pixels are repeated under a fixed 4-block Hadamard sign code, giving 2048 signal pixels, and the remaining 6144 pixels (75% of the input) are background at value $0.5 + 0.2\,z_t$ with one $z_t \sim \mathcal{U}[-\sqrt{3}, \sqrt{3}]$ per frame shared by all background pixels. The network receives the flattened vector with no spatial structure. Rendering the blobs relative to the goal matters: in absolute coordinates the $\tanh$ saturates at typical goals ($\|s_\tau^\star\| \approx$ 3–10) and the state becomes unobservable exactly where it must be regulated.

A.5 Training Protocol

Both agents are two-hidden-layer MLPs of width 128 with LayerNorm and swish after each hidden layer, separate actor and critic, and a linear output head; the actor's log standard deviation is a bounded global parameter. PPO uses 600 iterations per task with 512 environments, 4 epochs of 16 minibatches, clip 0.2, GAE $\lambda = 0.95$, value coefficient 0.5, entropy coefficient 0.01, and per-group gradient clipping. All mitigations are on for both optimizers: reward normalization by the running standard deviation of the discounted return, a regenerative ℓ_2 penalty pulling the LayerNorm affine parameters to their initialization, ℓ_2 of 10^{-2} on both output heads, decoupled weight decay, and a floor of -2 on the log standard deviation. Under Skewon the hidden weight matrices of both networks are constrained to the Stiefel manifold while biases, LayerNorm parameters, output heads and the log standard deviation are trained with AdamW.

Hyper-parameters are chosen per chain. For each nominal $|v|$ we run a 10-task sweep (5 hops, 3 seeds) and select the configuration with the fewest hops on which the training score fails to reach 0.8, breaking ties by mean H-task score and then by iterations to reach 0.8. The AdamW grid is learning rate $\{3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}\}\times$ weight decay $\{10^{-4}, 10^{-2}\} \times \beta_1 \in \{0.9, 0.5\}$ (12 configurations; extended to 54 with weight decay 10^{-3}, $\beta_1 = 0.75$ and LayerNorm penalty 10^{-3} for the fixed chains), and the Skewon grid is manifold learning rate $\{10^{-3}, 3 \cdot 10^{-3}\}\times$ momentum $\{0.5, 0.9\}$ (4 configurations). For the random chains the sweep is run on seeds 3–5 and the chain on seeds 0–2, so the winner is selected on different draws of $|v|$ than it is evaluated on. AdamW's winner was learning rate 10^{-4} with $\beta_1 = 0.5$ in 18 of the 24 cells; Skewon's was learning rate 10^{-3} with momentum 0.9 in every cell.

The normalized return $(\mathcal{R} - \mathcal{R}_0)/(\mathcal{R}^\star - \mathcal{R}_0)$, where $\mathcal{R}^\star$ is the return of the Riccati policy and $\mathcal{R}_0$ that of the feed-forward action with no feedback, evaluated on 256 deterministic episodes at the end of the task; under ths sclaing 1 is optimal and 0 is the constant policy. The input-output Jacobian norm is $\|\partial\pi(o)/\partial o\|_F$ of the actor mean, averaged over 32 observations from the policy's own rollout for 3 seeds of the chain.

B Experimental Details of MuJoCo-continual Experiments

All four variants share one architecture, one PPO configuration and one task schedule; the three methods differ only in which matrices are constrained and in how those matrices are updated. Observations are 84×84 RGB renders of the MuJoCo scene, simulated with the spring backend, with an action repeat of 4 and the last 4 frames stacked channel-wise, so the policy input is $84 \times 84 \times 12$. The encoder is a convolutional trunk followed by a CRATE feed-forward block of width 512 (Yu et al., 2023; Tiwari & Konidaris, 2022), a linear projection and a $\tanh$ output scaled by 0.5; actor and critic are separate two-hidden-layer heads of the same CRATE form. There is LayerNorm after every feedforward layer. Pixels are normalised inside the network, and the actor's log standard deviation is a state-dependent parameter initialised at -2 and clamped to $[-5, -1.4]$.

Each task runs for 5 million frames and a chain segment for 25 million, so the Ant-Fluid chains are 10 tasks over 50 million frames, Ant-Friction 15 tasks over 75 million, and the two humanoid variants 20 tasks over 100 million. PPO collects 10 steps from each of 128 environments per iteration, giving a batch of 1280 transitions split into 32 minibatches of 40 for each of 4 epochs, with clip 0.1, GAE $\lambda = 0.95$, discount 0.99, value coefficient 0.5, entropy coefficient 0, and gradient clipping at 0.05. Learning rates are constant; no annealing is used anywhere. Every chain warm-starts each segment from the final checkpoint of the previous one, and each chain is warm-started from a base trained with its own optimizer.

The three methods are distinguished by their treatment of six matrices: the encoder's CRATE block and its linear projection, both 512×512, and the hidden matrices of the actor and critic heads. Under Skewon these six are constrained to the Stiefel manifold and updated by the closed-form retraction of Section 3.2, while biases, LayerNorm parameters and the output heads are trained with AdamW. Under Parseval the same six matrices are left unconstrained but carry the orthogonality penalty of Chung et al. (2024), with the output heads excluded as in that work; the penalty is width-normalised against a reference width of 64. Under AdamW all matrices including those six are trained with decoupled weight decay and no constraint. In all three cases the encoder and the remaining parameters are trained with AdamW at the base learning rate, so the comparison isolates the treatment of the six matrices rather than the optimizer as a whole.

Hyper-parameters were selected on the two Ant variants and transferred unchanged to the humanoid variants, with the single exception of the base learning rate, which is lowered from $5 \cdot 10^{-5}$ to $3 \cdot 10^{-5}$ for the humanoid because the higher value is unstable there. For the two ant variants we ablate the frame stack: $\{2, 3, 4\}$, action repeats: $\{4, 5, 8\}$, Gradient clipping: $\{0.01, 0.02, 0.05\}$. For the AdamW variant we tried different weight decays: $\{10^{-5}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$. Table 3 lists the values that matter for the comparison; every entry is identical across all ten sequences of a variant.

C Experimental Details of ManiSkill Experiments

Both manipulation chains are simulated in ManiSkill 3 (Tao et al., 2024) on the SAPIEN engine (Xiang et al., 2020) and trained with the same PPO implementation and the same network form as the locomotion experiments, so the only differences from Section B are the observation space, the control mode and the task schedule. The policy sees rendered camera images together with a purely proprioceptive state vector; no privileged information about object or goal position enters the state, so every quantity the policy conditions on is available on a real arm. For the SO-100 the observation is two camera views, one fixed and one wrist-mounted, in `rgb+segmentation` mode alongside the six joint encoders, and control is joint-target delta position. For the Panda the observation is a single base camera in `rgb` mode alongside an eighteen-dimensional proprioceptive state, and control is joint delta position. The encoder is the same convolutional trunk with a CRATE block, a linear projection and a $\tanh$ output scaled by 0.5; the image and state branches are fused by concatenation, and actor and critic are separate CRATE heads.

Each rung of a chain is trained for 10 million frames and warm-started from the final policy of the previous rung, so the observation and action interfaces are identical along a chain and a checkpoint loads without modification. The SO-100 chain is five tasks, grasp, place, stack, cup-align and sort, and the Panda chain is four, grasp, place, stack and pyramid. We report four seeds per method on the SO-100 chain and three on the Panda chain, and success is the fraction of evaluation episodes in

	Skewon	Parseval	AdamW
Shared			
Frame stack		4	
Action repeat		4	
Parallel environments		128	
Rollout length		10	
Batch size		1280	
Minibatch size (32 minibatches)		40	
Update epochs		4	
Gradient clipping		0.05	
Base learning rate (ant)		$5 \cdot 10^{-5}$	
Base learning rate (humanoid)		$3 \cdot 10^{-5}$	
Method-specific			
Constrained-matrix learning rate	10^{-3}	—	—
Encoder manifold learning rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	—
Momentum	0	$\beta_1 = 0.9$	$\beta_1 = 0.9$
Weight decay	10^{-4} †	10^{-4}	10^{-4}
Parseval penalty coefficient	—	10^{-3}	—
Penalty target scale	—	2.0	—
Penalty reference width	—	True (64 base)	—

Table 3: Hyper-parameters for the MuJoCo-continual experiments. Entries above the rule are shared by all three methods and all four variants; entries below it apply only to the method that uses them. † it applies only to the unconstrained parameters, since the constrained matrices are on the Stiefel manifold by the retraction and are not decayed.

Variant	Method	SPS	Wall-clock (h)	Slowdown
Ant-Friction	AdamW	1699	4.10	1.00×
	Parseval	1664	4.19	1.02×
	Skewon	842	8.26	2.02×
Humanoid-Joint-Damping	AdamW	1618	4.31	1.00×
	Parseval	1397	5.25	1.16×
	Skewon	1185	5.88	1.37×

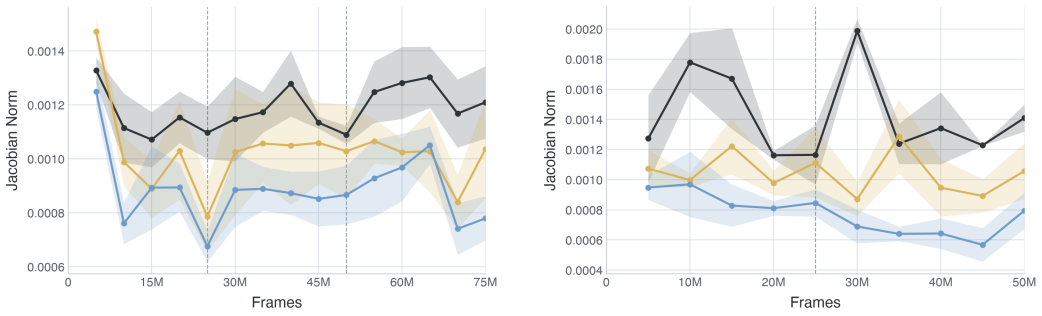
Table 4: **Training throughput**, measured on a single H100 over one 25 million frame chain segment and averaged over all runs of each variant. SPS is environment steps per second; slowdown is relative to AdamW on the same variant. The constrained update costs one matrix sign evaluation per constrained matrix per iteration, which is more expensive on the ant, where rendering and physics are cheaper and the optimizer is a larger share of the step.

which the task is completed at any point in the episode. The from-scratch controls in Table 1 use an identical configuration but train each task from random initialisation with no warm start.

As in the locomotion experiments the methods differ only in the treatment of the encoder's CRATE block and projection and the hidden matrices of the two heads. Under Skewon these are constrained to the Stiefel manifold and updated by the closed-form retraction; under AdamW they are unconstrained and trained with decoupled weight decay. The remaining parameters are trained with AdamW at the base learning rate in both cases. One task-level exception is made to the shared configuration: the entropy coefficient is lowered from 0.01 to 0.003 on the Panda pyramid task for both methods, because at 0.01 the policy entropy diverges on that task and neither method learns it. Table 5 lists the values that matter for the comparison.

C.1 Task Descriptions

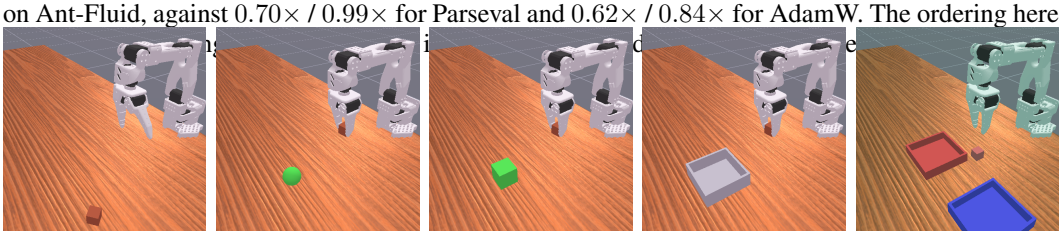
The five tasks with the SO 100 arm are described in detail below:



(a) Ant-Friction, 15 tasks

(b) Ant-Fluid, 10 tasks

Figure 7: Frobenius norm of the input–output Jacobian $\|J\|_F = \|\partial F/\partial x\|_F$ of the policy, measured at the start of every task from on-policy rollouts under that task's own dynamics coefficients (256 samples per checkpoint, five sequences per method, mean $\pm$ one standard error). Dashed vertical lines mark the 25M-frame training segments. Skewon is the only method whose Jacobian norm does not contract over the chain, retaining $0.91\times$ its initial value on Ant-Friction and growing to $1.11\times$ on Ant-Fluid, against $0.70\times$ / $0.99\times$ for Parseval and $0.62\times$ / $0.84\times$ for AdamW. The ordering here

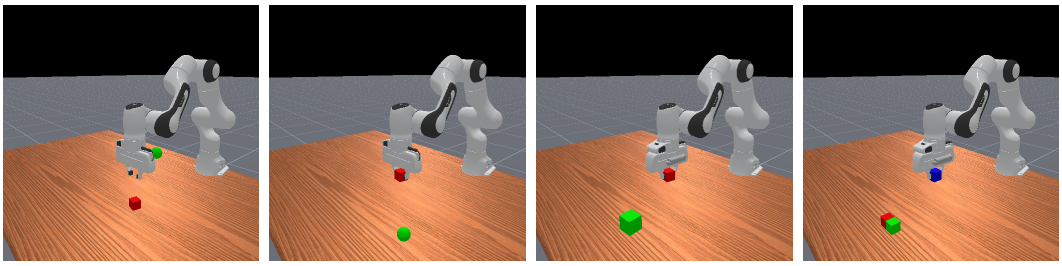


(a) Grasp (b) Place (c) Stack (d) Cup-Align (e) Sort

Figure 8: The five tasks of the SO-100 manipulation chain, shown at the start of an episode. Each task is warm-started from the policy of the task before it, and every task after the first begins with the object already held, so the observation and action interfaces are identical along the chain. In Sort the object is a cube or a ball with equal probability and its identity is withheld from the state vector, so the arm must infer which bin to use from the camera alone.

1. **Grasp:** the arm must reach a cube resting on the table and close its gripper on it, succeeding when a stable grasp is established. This is the only task in the chain that begins with an empty gripper.
2. **Place:** the episode begins with the cube already grasped, and a goal marker is sampled at a random position on the table. The arm must carry the cube to the marker and release it there, succeeding when the cube comes to rest within a small distance of the goal.
3. **Stack:** a second cube is placed on the table at the sampled target and the arm must stack the carried cube on top of it. Success requires both precision, since the supporting surface is small, and stability, since the tower must not topple after release. The base cube is visible only in the camera images and is not included in the proprioceptive state.
4. **Cup-align:** the flat goal marker of the place task is replaced by a walled bin, and the arm must deposit the grasped cube inside it. Success requires the cube to lie within the opening and below the rim once released, so the arm must clear the walls rather than approach the goal along the table surface.
5. **Sort:** each episode presents either a cube or a ball, held at the start, and two fixed bins of different colours are placed on the table, one assigned to each shape. The identity of the held object is withheld from the state vector, so the policy must infer which object it is holding from the camera images alone and carry it to the matching bin.

The four tasks with the Franka Panda arm are described in detail below:



(a) Grasp (b) Place (c) Stack (d) Pyramid

Figure 9: The four tasks of the Franka Panda manipulation chain, shown at the start of an episode. As in the SO-100 chain each task is warm-started from the policy of the task before it, and every task after the first begins with the object already held. In Stack the base cube is enlarged relative to the carried cube to give a more forgiving landing surface; in Pyramid two cubes are pre-placed side by side and the held cube must be set down on the seam between them.

1. **Grasp:** the arm must reach the cube resting on the table and close its gripper on it, succeeding once the cube is grasped and lifted clear of the surface. As with the SO-100, this is the only task that begins with an empty gripper, and it is given a shorter horizon than the tasks that follow.
2. **Place:** the episode begins with the cube already held at the gripper, and a goal position is sampled in the air above the table. The arm must transport the cube to the goal and hold it there, so the task rewards carrying and maintaining the grasp rather than releasing.
3. **Stack:** a second cube is placed on the table within reach and the arm, starting from the grasped pose, must stack the carried cube on top of it. Success requires the arm to lower the cube onto the base, release it, and leave it at rest, so unlike the place task the grasp must eventually be given up at the right moment.
4. **Pyramid:** two cubes are pre-placed side by side on the table to form a base pair, and the episode begins with a third cube held at the gripper. The arm must stack the held cube on the apex formed between the two, which requires finer positioning than the single-cube stack because the supporting surface is the seam between two objects.

	SO-100		Panda	
	Skewon	AdamW	Skewon	AdamW
Parallel environments	128		256	
Rollout length	50		64	
Batch size	6400		16384	
Minibatch size (32 minibatches)	200		512	
Update epochs	4			
Frames per task	$10 \cdot 10^6$			
Base learning rate	$3 \cdot 10^{-4}$			
Discount γ	0.95		0.8	
GAE λ	0.9			
Clip	0.2			
Entropy coefficient	0.01		0.01 †	
Gradient clipping	0.5			
Constrained-matrix learning rate	10^{-3}	—	10^{-3}	—
Encoder manifold learning rate	$5 \cdot 10^{-4}$	—	$5 \cdot 10^{-4}$	—
Weight decay	—	—	0	10^{-4}

Table 5: Hyper-parameters for the ManiSkill manipulation chains. Learning rates are constant, with no annealing. †The entropy coefficient is 0.003 rather than 0.01 on the Panda pyramid task, for both methods, this change was the only effective change in making it approach a satisfactory success rate.

Algorithm 1 Gram-Skewon (one weight matrix; $\mathrm{NS}_k = k$ Newton–Schulz / Polar-Express steps)

Require: initial weight $W_0 \in \mathbb{R}^{m\times n}$ of full column rank; iterations T; step size $\eta > 0$; momentum β; conditioning cap $\kappa_{\max}$; sign iterations k
Ensure: iterates W_t with $W_t^\top W_t = G_0$ for every t

$U\,\mathrm{diag}(s)Q^\top \leftarrow \mathrm{svd}(W_0)$ ▷ once; $O(mn^2)$
$s \leftarrow \max\big(s,\, s_{\max}/\kappa_{\max}\big)$ ▷ spectrum floor, Eq. equation 10
$V_0 \leftarrow UQ^\top, \quad P \leftarrow Q\,\mathrm{diag}(s)Q^\top, \quad G_0 \leftarrow P^2, \quad M_{-1} \leftarrow 0$
for $t = 0, 1, \ldots, T-1$ **do**
  $\mathcal{G}_t \leftarrow \nabla_W \mathcal{L}(V_t P)$ ▷ Euclidean gradient, tall form
  $M_t \leftarrow \beta M_{t-1} + \mathcal{G}_t, \quad \widetilde{\mathcal{G}}_t \leftarrow \mathcal{G}_t + \beta M_t$ ▷ ambient Nesterov momentum
  $N_t \leftarrow \mathrm{skew}\big(\widetilde{\mathcal{G}}_t P V_t^\top\big)$ ▷ $= \mathrm{skew}(\widetilde{\mathcal{G}}_t W_t^\top)$; $O(m^2 n)$
  $Y_t \leftarrow -\,\mathrm{skew}\big(\mathrm{NS}_k(N_t)\big)$ ▷ $\approx -\,\mathrm{msign}(N_t)$, the LMO solution; $O(km^3)$
  $B_t \leftarrow Y_t V_t$ ▷ update direction, $\|B_t\|_{\mathrm{rot}} \le 1$; $O(m^2 n)$
  $V_{t+1} \leftarrow \mathrm{qr}\big(V_t + \eta\, B_t\big)$ ▷ retraction onto $\mathrm{St}(m,n)$; $O(mn^2)$
  $W_{t+1} \leftarrow V_{t+1} P$ ▷ $W_{t+1}^\top W_{t+1} = P V_{t+1}^\top V_{t+1} P = G_0$
end for

D Gram Skewon Derivation

Here we derive the algorithm presented above step by step.

The tangent space is unchanged. Differentiating the constraint in equation 2 gives $A^\top W + W^\top A = 0$ — the same equation as for Stiefel, with no dependence on G_0. This yields the fact the method rests on.

Lemma 1. *For any W of full column rank, $T_W\mathcal{M}_{G_0} = \{\, YW \;:\; Y \in \mathrm{Skew}(m) \,\}$.*

Proof. Throughout, let $\Psi : \mathrm{Skew}(m) \to \mathbb{R}^{m\times n}$, $\Psi(Y) = YW$, which is linear, and let $h : \mathbb{R}^{m\times n} \to \mathrm{Sym}(n)$, $h(W) = W^\top W - G_0$, so that $\mathcal{M}_{G_0} = h^{-1}(0)$. Note first that every $W \in \mathcal{M}_{G_0}$ automatically has full column rank: $W^\top W = G_0 \succ 0$ is invertible, so $Wz = 0$ forces $G_0 z = 0$ and hence $z = 0$. We prove the two sets are equal by showing (i) the inclusion $\Psi(\mathrm{Skew}(m)) \subseteq T_W\mathcal{M}_{G_0}$, and (ii) that both are linear subspaces of $\mathbb{R}^{m\times n}$ of the same finite dimension. Since a subspace of a finite-dimensional space that is not the whole space has strictly smaller dimension, (i) and (ii) together give equality; neither alone would suffice.

Step 1: $\mathcal{M}_{G_0}$ is an embedded submanifold and $T_W\mathcal{M}_{G_0} = \ker Dh(W)$. The differential of h at W is $Dh(W)[A] = A^\top W + W^\top A$, which indeed lands in $\mathrm{Sym}(n)$ because $(A^\top W + W^\top A)^\top = W^\top A + A^\top W$. It is surjective onto $\mathrm{Sym}(n)$: given any $S \in \mathrm{Sym}(n)$, put $A = \frac{1}{2} W G_0^{-1} S$, which is well defined since G_0 is invertible. Using $S^\top = S$ and $G_0^{-\top} = G_0^{-1}$,

$$A^\top W \;=\; \tfrac{1}{2} S^\top G_0^{-\top} W^\top W \;=\; \tfrac{1}{2} S G_0^{-1} G_0 \;=\; \tfrac{1}{2} S, \qquad W^\top A \;=\; \tfrac{1}{2} W^\top W G_0^{-1} S \;=\; \tfrac{1}{2} G_0 G_0^{-1} S \;=\; \tfrac{1}{2} S,$$

so $Dh(W)[A] = S$. Hence 0 is a regular value of h, $\mathcal{M}_{G_0}$ is an embedded submanifold, its tangent space at W is the kernel of the differential, and

$$\dim T_W\mathcal{M}_{G_0} \;=\; \dim \mathbb{R}^{m\times n} - \dim \mathrm{Sym}(n) \;=\; mn - \frac{n(n+1)}{2}. \tag{4}$$

Step 2: the inclusion. Let $Y \in \mathrm{Skew}(m)$. Then

$$Dh(W)[YW] \;=\; (YW)^\top W + W^\top (YW) \;=\; W^\top Y^\top W + W^\top Y W \;=\; W^\top (Y^\top + Y) W \;=\; 0,$$

since $Y^\top = -Y$. So $\Psi(Y) \in \ker Dh(W) = T_W\mathcal{M}_{G_0}$.

Step 3: the kernel of Ψ. Write $R = \mathrm{range}(W)$, which has $\dim R = n$ by full column rank, and let $E \in \mathrm{St}(m,n)$ and $E_\perp \in \mathrm{St}(m, m-n)$ be orthonormal bases of R and $R^\perp$, so $[\,E \mid E_\perp\,]$ is orthogonal. Suppose $YW = 0$ with Y skew. The columns of W span R, so $Yv = 0$ for all $v \in R$; and then for arbitrary $u \in \mathbb{R}^m$ and $v \in R$,

$$\langle Yu,\, v\rangle \;=\; -\langle u,\, Yv\rangle \;=\; 0,$$

so $\mathrm{range}(Y) \subseteq R^{\perp}$ as well. In the block form induced by $[\,E \mid E_{\perp}\,]$, these two statements say exactly that the first block column of Y vanishes, $E^{\top}YE = 0$ and $E_{\perp}^{\top}YE = 0$; skewness then forces the first block row to vanish too, $E^{\top}YE_{\perp} = -(E_{\perp}^{\top}YE)^{\top} = 0$. Hence

$$Y \;=\; [\,E \mid E_{\perp}\,] \begin{pmatrix} 0 & 0 \\ 0 & Z \end{pmatrix} [\,E \mid E_{\perp}\,]^{\top}, \qquad Z \in \mathrm{Skew}(m-n),$$

and conversely every such Y is skew and satisfies $YW = 0$. Therefore $\ker \Psi \cong \mathrm{Skew}(m-n)$ and $\dim \ker \Psi = \frac{(m-n)(m-n-1)}{2}$.

Step 4: dimensions agree. By rank–nullity applied to Ψ,

$$\begin{aligned} \dim \Psi(\mathrm{Skew}(m)) \;&=\; \frac{m(m-1)}{2} - \frac{(m-n)(m-n-1)}{2} \\ &= \frac{\big(m^2 - m\big) - \big(m^2 - 2mn + n^2 - m + n\big)}{2} \\ &= \frac{2mn - n^2 - n}{2} \;=\; mn - \frac{n(n+1)}{2}, \end{aligned}$$

which equals $\dim T_W \mathcal{M}_{G_0}$ by equation 4. With Step 2 this proves the lemma. □

Lemma 1 states the parametrisation of the tangent space by a skew matrix, the object Skewon's closed form is built on, survives verbatim at a non-orthonormal W. What does not survive is the *metric*: the map $Y \mapsto YW$ is an isometry for the spectral norm only when $W^{\top}W = I$. This is the one place where the pretrained case genuinely differs, and we address it in the choice of metric below rather than by abandoning the closed form.

The rotation metric. An update rule also needs to define a norm to say how large a step is. We use the norm inherited from the group that acts on the manifold. Observe that $\mathcal{M}_{G_0}$ is exactly an orbit of the orthogonal group acting on the left: if W and W_0 share a Gram then, by equation 8 below, they share the factor P and differ only in their orthonormal factor, so

$$\mathcal{M}_{G_0} \;=\; \big\{\, QW_0 \;:\; Q \in O(m) \,\big\}. \tag{5}$$

Preserving the Gramian *is* rotating the columns rigidly in $\mathbb{R}^m$, and by Lemma 1 the skew matrices Y are precisely the generators of that action: for any $Y \in \mathrm{Skew}(m)$ the curve $t \mapsto \exp(tY)W$ satisfies $\big(\exp(tY)W\big)^{\top} \exp(tY)W \;=\; W^{\top}W \;=\; G_0$ for every t, so it never leaves $\mathcal{M}_{G_0}$. We therefore measure a tangent vector by the size of its generator rather than by its own size, defining the *rotation norm*

$$\|A\|_{\mathrm{rot}} \;=\; \min\big\{\, \|Y\|_2 \;:\; A = YW,\; Y \in \mathrm{Skew}(m) \,\big\}, \qquad A \in T_W \mathcal{M}_{G_0}. \tag{6}$$

The name is literal: the eigenvalues of a skew Y are $\pm i\theta_k$ with $\theta_k \le \|Y\|_2$, so a unit-norm generator rotates each invariant 2-plane by at most one radian, and $\|A\|_{\mathrm{rot}} \le \eta$ means the step travels at most η radians in $O(m)$. When $W^{\top}W = I$ the map $Y \mapsto YW$ is an isometry and $\|\cdot\|_{\mathrm{rot}}$ coincides with the spectral norm $\|A\|_2$; this equality is what Skewon's closed form relies on, and it is exactly what fails at a pretrained W, where the two norms differ by up to $\kappa(W)$.

The update direction. Let $\mathcal{G} = \partial \mathcal{L}/\partial W$ be the Euclidean gradient. We take the update to be the following minimisation problem over the tangent space measured in the rotation norm equation 6, that is, over $\{A \in T_W \mathcal{M}_{G_0} : \|A\|_{\mathrm{rot}} \le 1\}$:

$$Y^{\star} \;=\; \operatorname*{arg\,min}_{Y \in \mathrm{Skew}(m),\; \|Y\|_2 \le 1} \langle \mathcal{G},\, YW \rangle \;=\; -\,\mathrm{msign}\big(\mathrm{skew}(\mathcal{G}W^{\top})\big), \qquad \Delta W \;=\; \eta\, Y^{\star} W. \tag{7}$$

The closed form follows because $\langle \mathcal{G}, YW \rangle = \langle \mathcal{G}W^{\top}, Y \rangle = \langle \mathrm{skew}(\mathcal{G}W^{\top}), Y \rangle$ for skew Y, and msign of a skew matrix is skew, so the minimiser is feasible. Equation equation 7 is Skewon's step evaluated at a pretrained W; we verified it against a convex solver at $\kappa(W) = 1, 3, 10, 50$ and recover the same optimum in every case.

Two properties make $\|\cdot\|_{\mathrm{rot}}$ the right budget here. First, it bounds the *relative* functional change uniformly over inputs. For any x, with $y = Wx$,

$$\Delta W x \;=\; \eta\, Y^{\star}(Wx) \;=\; \eta\, Y^{\star} y \qquad \Longrightarrow \qquad \max_{x \neq 0} \frac{\|\Delta W x\|}{\|Wx\|} \;=\; \eta,$$

since $\|Y^\star\|_2 = 1$: every input direction moves by at most η *as a fraction of its own output norm*. A spectral trust region $\|\Delta W\|_2 \le \eta$ bounds only the absolute change, so the same budget is a far larger relative perturbation along directions with small σ — precisely the fragile, near-singular directions of a pretrained checkpoint. Second, the budget is scale-free: one global η means the same relative change in every layer irrespective of $\|W\|$, which across `smolvla_base` varies by more than an order of magnitude ($\sigma_{\max}$ from 0.05 to 3.7). A Euclidean trust region cannot do either without per-layer tuning.

Retraction without inverses, via a one-time polar split. Applying equation 3 to W directly would require retracting onto $\mathcal{M}_{G_0}$, which in general needs $G_0^{1/2}$ or a Cayley transform. We avoid this by factoring the constraint once. Write the polar decomposition

$$W_0 \;=\; V_0 P, \qquad V_0 = \mathrm{msign}(W_0) \in \mathrm{St}(m,n), \qquad P = V_0^\top W_0 = (W_0^\top W_0)^{1/2}, \tag{8}$$

so that $\mathcal{M}_{G_0} = \{VP : V \in \mathrm{St}(m,n)\}$ with $G_0 = P^2$. Since $YW = (YV)P$, the Gram-Skewon step on W *is* the ordinary Skewon step on the orthonormal factor V, carrying the pulled-back gradient

$$\mathcal{G}_V \;=\; \mathcal{G}P, \qquad \text{because} \qquad \langle \mathcal{G},\, \Delta V\, P\rangle = \langle \mathcal{G}P,\, \Delta V\rangle, \tag{9}$$

and the direction is unchanged, since $\mathrm{skew}(\mathcal{G}_V V^\top) = \mathrm{skew}(\mathcal{G}PV^\top) = \mathrm{skew}(\mathcal{G}W^\top)$ by symmetry of P. All learning happens in V; the factor P is computed once and never updated, never inverted. The constraint then holds by construction rather than by correction,

$$W^\top W \;=\; PV^\top VP \;=\; P^2 \;=\; G_0 \quad \text{exactly.}$$

Retracting V rather than W is also numerically cheaper: $V + \eta YV$ is near-orthonormal, so the Newton–Schulz retraction converges in two iterations at any conditioning, whereas retracting $W + \eta YW$ needs 7 to 17 iterations as κ grows from 3 to 200.

Conditioning: the manifold freezes the spectrum, so it must be chosen once. Because P is fixed, the singular values of W are fixed for the entire run — including any near-null directions the pretrained checkpoint happens to contain. Freezing those would both preserve a degenerate subspace forever and make every msign slow and inaccurate in low precision. We therefore apply a one-time spectrum floor when forming P: from the SVD $W_0 = U\,\mathrm{diag}(s)Q^\top$, set

$$s_t = \max\!\big(s,\; s_{\max}/\kappa_{\max}\big), \qquad V_0 = UQ^\top, \qquad P = Q\,\mathrm{diag}(s_t)Q^\top, \qquad W \leftarrow V_0 P. \tag{10}$$

The resulting edit is small, since V_0 is an isometry and hence $\|W_{\text{new}} - W_0\|_F / \|W_0\|_F = \|s_t - s\|/\|s\|$, which is supported only on directions carrying almost no energy. For `smolvla_base`, 96 of the 117 trainable matrices already satisfy $\kappa \le 33$; the exceptions are the cross-attention key/value projections (κ from 287 to 1.3×10^4) and `action_time_mlp_out` ($\kappa \approx 1.9 \times 10^4$), all of which are layers created fresh for the action expert rather than inherited from the pretrained VLM. At $\kappa_{\max} = 30$ the floor changes those matrices by 1–2% in Frobenius norm and leaves the rest untouched. After initialisation, $\kappa \le \kappa_{\max}$ holds for the whole task chain by construction — with no regulariser, no weight decay, and no tuning.

The algorithm. Putting these all together, one step on one matrix is one msign per matrix per step, three matrix products, and no inverse, no Cayley transform, and no eigendecomposition after initialisation. The momentum buffer M is *ambient*: it accumulates the Euclidean gradient in W-space, before the pull-back and before the sign, and is not parallel-transported between tangent spaces (we use $\beta = 0.95$ with the Nesterov form, as in Muon). Since the direction is recomputed by msign at every step, the buffer only smooths which tangent direction is selected; it never moves the iterate off $\mathcal{M}_{G_0}$, because the update is applied to V alone. When $m \ge 2n$ the $m \times m$ sign can be replaced by a QR of $[\,V \mid \mathcal{G}_V\,]$ followed by a $2n \times 2n$ sign, reducing the cost from $O(m^3)$ to $O(mn^2)$. Memory is one fp32 copy of V plus the $n \times n$ factor P, roughly $1\times$ the parameter count against AdamW's $2\times$. Numerically we observe the Gram held to a relative error of 10^{-6} and orthonormality of V to 10^{-8} over a full run.

E Experimental Details of Real World Experiments

E.1 Real World Task Decriptions

The task description input to the VLA

1. **Pick** (*"Pick up the cube"*; 200 demonstrations, 15.6 s mean). A single cube rests on the table at a position and colour that vary between episodes. The arm starts at a fixed rest pose with the gripper closed and must approach the cube, open the jaws, close on it, and lift it clear of the surface. This is the only task that requires prehension; the remaining four begin with the cube already grasped.
2. **Place** (*"Place the cube on the black spot"*; 104 demonstrations, 10.1 s mean). The episode begins with the cube held in the gripper. The arm must transport it to a black disc marked on the table and release it on the target.
3. **Stack** (*"Stack the cube on top of the blue cube"*; 200 demonstrations, 12.6 s mean). Starting with a cube in the gripper, the arm must align it above a second, blue cube resting on the table and release it so that it remains balanced on top. Success requires finer vertical and lateral precision than Place, since the supporting face is the width of a single cube.
4. **Cup** (*"Place the cube in the blue cup"*; 100 demonstrations, 10.6 s mean). Starting with a cube in the gripper, the arm must carry it over a blue cup and release it inside. The target is a container rather than a surface, so the release must clear the rim.
5. **Sort** (six instructions, 240 demonstrations, 9.9 s mean). Two cups, one red and one green, are placed on the table together with a cube that is blue, yellow, or orange. The instruction names both the cube and the destination cup, giving six combinations of the form *"Put the {blue, yellow, orange} cube in the {red, green} cup"*, with 20 demonstrations each. Unlike the preceding tasks, the scene alone does not determine the goal: the same visual observation admits two correct behaviors, and only the language instruction disambiguates them.

E.2 Policy and Backbone

All real-world runs fine-tune **SmolVLA** (Shukor et al., 2025) through the LeRobot framework (Cadene et al., 2026). The backbone is the SmolVLM2-500M-Video-Instruct vision-language model, of which the first 16 transformer layers are used. Attached to it is an *action expert* whose hidden width is $0.75\times$ the VLM width; the expert attends to the VLM through cross-attention, with a self-attention layer interleaved every second layer.

Observations at each step are two 640×480 RGB streams at 30 Hz — one fixed exocentric camera and one wrist camera — resized with padding to 512×512, the 6-DoF joint state of the SO-101 follower arm, and the task string, tokenized to at most 48 tokens. A single observation step is used ($n_{\text{obs}} = 1$); no observation history is stacked. State and action vectors are zero-padded from 6 to the model's 32 dimensions. Images are passed through unnormalized; state and action channels are normalized to zero mean and unit variance.

The policy emits an action *chunk* of 50 future joint commands, trained with the flow-matching objective of Shukor et al. (2025): a noise tensor and a time t are sampled, the noise and the ground-truth chunk are interpolated, and the network regresses the velocity field connecting them. At inference the chunk is recovered by integrating that field over 10 steps, and all 50 predicted actions are executed before the next chunk is requested.

E.3 Which Parameters Are Updated

We fine-tune in the *expert-only* regime, which is the configuration that isolates the optimizer comparison from confounds in the pretrained representation. Concretely:

- The vision encoder is frozen and held in evaluation mode.
- The entire VLM — vision *and* language networks — is frozen and held in evaluation mode, so its BatchNorm/dropout statistics do not drift between stages.
- Only the action expert, together with the state and action projection layers, receives gradients.

This leaves 99.9 M trainable parameters in 155 tensors, of which 117 are matrices, out of roughly 450 M total. The distinction matters for Gram-Skewon, which acts on matrix-shaped parameters:

the 117 matrices are exactly the set carried on the Stiefel manifold, while the remaining 38 vector-shaped tensors (biases, norms) are updated by AdamW in both arms. Because the VLM is frozen throughout the chain, every stage inherits an identical vision-language representation and the only quantity that accumulates across tasks is the action expert — so any loss of plasticity we measure is a property of the expert, not of a drifting backbone.

Normalization statistics are computed once, on the first stage of the chain, and then held fixed for every subsequent stage. Recomputing them per stage would silently rescale the action space between tasks and make retention numbers incomparable.

E.4 Optimization Hyperparameters

Both arms share every setting except the optimizer itself:

Setting	Value
Batch size	32
Schedule	constant (no warmup, no decay)
Epochs per stage	10
Master weights	fp32
Seed	1000
Trainable parameters	99.9 M (action expert only)
AdamW	
Learning rate	1×10^{-4}
Weight decay	1×10^{-5}
(β_1, β_2)	(0.9, 0.95)
Gradient clip	10
Gram-Skewon	
Rotation budget	1×10^{-3}
$\kappa_{\max}$	30
Newton–Schulz steps	12
Momentum	0.95

A constant learning rate is used deliberately. A decaying schedule would make each stage's final update smaller than its first, confounding position-in-chain with effective step size, and the quantity under study is precisely how a stage's position affects what it overwrites.

The Gram-Skewon learning rate is not comparable to AdamW's: it is a rotation budget in radians applied to the orthonormal factor V, not a displacement in parameter space. The two values were selected independently, each as the largest that trained stably on the first stage. We note also that matrices carried on the manifold cannot receive weight decay — the constraint already pins $\|W\|$ — so the 1×10^{-5} decay is a regularizer present in the AdamW arm alone.

E.5 Chain Schedule

Each stage is trained for 10 epochs over its own dataset and initialized from the final checkpoint of the preceding stage; no data from earlier stages is replayed. Step counts follow from the frame counts at batch size 32:

Task	Episodes	Frames	Steps
Pick	200	77,554	24,250
Place	104	31,504	9,850
Stack	201	66,128	20,650
Cup	100	31,779	9,950
Sort	242	62,746	19,650
Total	847	269,711	84,350

At every evaluation we additionally record the action MAE of the current policy on a held-out sample from each *earlier* stage's dataset, which yields the forgetting curve directly from the training run.