LEARNING A SYNERGY BASIS FROM REWARD ALONE

Anonymous authors
Paper under double-blind review

ABSTRACT

Musculoskeletal control involves action spaces where muscles far outnumber the mechanical degrees of freedom they drive, creating a redundancy that challenges standard reinforcement learning (RL). A natural remedy drawn from motor neuroscience is to act through a smaller set of muscle synergies. Most existing synergy-based RL methods obtain their basis from a separate discovery phase, a dynamics model, or physiology rather than from the task reward. We introduce Plexus, a latent policy learning method that constrains actions to non-negative combinations of synergy vectors and makes the basis a free parameter of the actor. Crucially, this basis is learned end-to-end, from reward alone, at the same time as the actor. Because the decoder is learned and otherwise unconstrained, it is generically non-orthogonal and distorts volume, so latent and ambient entropy no longer represent the same quantity. This makes the basis lose rank and the policy steer a subspace narrower than its latent size advertises. We remedy this by measuring entropy on the induced action manifold, through a Gram determinant correction to the log-density. We show that when the decoder is jointly learned, letting this quantity flow through the decoder keeps the basis well conditioned. Across six MyoSuite tasks on two musculoskeletal models, Plexus matches or exceeds full-dimensional baselines on every task and is competitive with synergy-based methods at matched latent width, while yielding a single state-independent basis that can additionally be initialized or constrained with biomechanical priors, or inspected for interpretability.

1 INTRODUCTION

Biological movement relies on the coordinated activation of many muscles: a single human hand involves on the order of 30 muscles, a leg over 60, and the full body more than 600, far exceeding the mechanical degrees of freedom they control. This overactuation means that infinitely many muscle activation patterns can produce the same joint torques (Bernstein, 1967). Motor neuroscience has long argued that the central nervous system copes with this redundancy by recruiting small sets of muscle synergies (Tresch et al., 1999; d'Avella et al., 2003; Bizzi et al., 2008), reducing the effective dimensionality of motor commands by activating muscles in a limited number of stereotyped patterns whose combination produces the full activation vector.

This redundancy poses significant challenges for reinforcement learning (RL) in musculoskeletal simulators, which expose large muscle action spaces for tasks lying on a much lower-dimensional manifold. Training in such environments is difficult for two reasons: (i) the same reward can arise from countless co-activation patterns, leaving no unique target, and (ii) exploration in large action spaces is prohibitively expensive. How an agent trades this exploration against exploitation strongly influences its performance. Maximum-entropy RL (Section 2.1) directly implements this in the objective by augmenting the reward with the entropy of the policy, so that policies maintaining broader support are directly favored (Haarnoja et al., 2018b).

A complementary remedy is to shrink the space being explored. Rather than emitting one command per muscle, which scales poorly as the action dimensionality grows, the policy emits a low-dimensional latent action and a decoder expands it into a full activation vector. While this might simplify the search, it also complicates the term that was driving it. Indeed, unless the decoder preserves volume, a latent policy offers two different base measures in which entropy can be measured: the latent space it samples from, and the action space it acts in.

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.

We resolve this by a change of base measure. Entropy should be measured as volume on the manifold the policy actually occupies rather than in the latent space it is sampled from. With a learned decoder particularly, the correction is differentiable with respect to the decoder itself. So, beyond repairing the entropy estimate, it also can shape the decoder. Our contributions are:

- *An action manifold learned from reward alone.* We make the decoder a parameter of the actor, learned end-to-end from reward alone, jointly with the policy. We test it across six MyoSuite tasks spanning individual joint control, manipulation and locomotion.
- *The impact and effects of the Jacobian correction.* We show the Gram-determinant volume element keeps the basis well conditioned, preventing rank collapse. We isolate the term's gradient into the decoder as what maintains the conditioning.
- *An interpretable architecture and its affordances.* We learn one fixed non-negative matrix, which can independently be: inspected, initialized from a basis found elsewhere, held frozen or constrained by some biomechanical prior.

2 Background

2.1 Maximum-entropy reinforcement learning

We consider an infinite-horizon Markov decision process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A} \subseteq \mathbb{R}^A$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in (0, 1)$ (Sutton & Barto, 2018). Maximum-entropy (MaxEnt) RL is closely related to the maximum-entropy framework for inverse RL (Ziebart et al., 2008) and was cast as an explicit RL objective via soft Q-functions (Haarnoja et al., 2017), leading to the entropy-augmented return (Haarnoja et al., 2018b):

$$\max_{\pi} \; J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \big(r(s_t, a_t) + \alpha\, \mathcal{H}(\pi(\cdot|s_t))\big)\right], \tag{1}$$

where $\alpha > 0$ is the entropy temperature and $\mathcal{H}(\pi(\cdot|s)) = -\mathbb{E}_{a\sim\pi}[\log \pi(a|s)]$. The entropy bonus encourages stochastic, diverse policies that are more robust to environment shifts and model mismatch (Eysenbach & Levine, 2022).

Soft Actor-Critic (SAC) (Haarnoja et al., 2018b), which we use as a base in this work, is an off-policy actor-critic algorithm optimizing this objective in continuous action spaces. The actor minimizes

$$\mathcal{L}_{\pi} = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi(\cdot|s)}\big[\alpha \log \pi(a|s) - Q(s, a)\big], \tag{2}$$

where Q is a learned soft Q-function and $\mathcal{D}$ a replay buffer. The temperature is usually not fixed but tuned by dual gradient descent against a target entropy $\bar{\mathcal{H}}$ (Haarnoja et al., 2018c), so the same log-density that appears in Eq. 2 also determines α.

2.2 Change of variables and the entropy of a transformed policy

Both terms in Eq. 2 require a density for the action actually sent to the environment, which in practice is the image of a sample from a simpler distribution. In SAC the policy is parameterized as a squashed Gaussian: a sample $u \sim \mathcal{N}(\mu(s), \sigma^2(s))$ is passed through an elementwise $\tanh$, and the log-density is corrected by the change-of-variables formula (Haarnoja et al., 2018c):

$$\log \pi(a|s) = \log \pi_U(u|s) - \sum_{j=1}^{A} \log(1 - \tanh^2(u_j)). \tag{3}$$

When the environment expects actions in $[\ell, h]^A$, an additional affine rescaling $a_{\text{env}} = \frac{h-\ell}{2}\, a + \frac{h+\ell}{2}$ is applied, and its log-Jacobian $\sum_j \log \frac{h_j - \ell_j}{2}$ is subtracted from the log-density.

A change of variables requires rescaling the density by the determinant of the Jacobian of the transform. The correction in Eq. 3 is a sum over independent dimensions, a decomposition that is valid because the map $u \mapsto \tanh(u)$ and the rescaling are elementwise: their Jacobians are diagonal, so the log-determinant splits into per-dimension terms. The map is also square and invertible, so $\det J$ is well defined, non-zero and the image remains full-dimensional. When the action space is further transformed by a learned non-orthogonal decoder, the relationship between latent and action-space entropy requires additional care; we develop this in Section 4.2.

3 RELATED WORK

Musculoskeletal simulators and models. Musculoskeletal simulation has long been dominated by OpenSim (Seth et al., 2018), which provides anatomically accurate models validated against experimental data, but whose simulation speed scales poorly with model complexity, making it impractical for deep RL over millions of environment steps. The MuJoCo physics engine (Todorov et al., 2012) , increasingly used in RL research, offers fast and stable simulation with contact dynamics, but does not provide musculoskeletal simulation out-of-the-box. MyoSim (Wang et al., 2022) addresses this with an automated pipeline converting validated OpenSim models into MuJoCo, yielding physiologically accurate models orders of magnitude faster than their OpenSim counterparts. Building on this, MyoSuite (Caggiano et al., 2022a) packages these models into a suite of increasingly challenging motor control tasks, from simple posture control to contact-rich dexterous manipulation or locomotion. The suite has also served as the primary benchmark for musculoskeletal RL through the Myo-Challenge series (Caggiano et al., 2022b; 2023). In this work, we adopt it for its anatomical fidelity, speed, challenging environments and comparability with prior work.

Learning in overactuated action spaces. Several strategies deal with high-dimensional muscle action spaces without compressing them, through exploration, curricula, reward design or imitation. DEP-RL addresses exploration in overactuated systems through differential extrinsic plasticity (Schumacher et al., 2023), and Schumacher et al. (2025) extend it to locomotion on a 90-muscle model using biologically plausible rewards, imposing no synergy structure. Chiappa et al. (2024) show that curriculum-based RL discovers low-dimensional synergies without explicit bias, but that task performance depends on more dimensions than variance-based analysis suggests. Simos et al. (2026) analyze emergent synergies post hoc in a motion-imitation framework. Chiappa et al. (2023) introduce Lattice, which injects temporally correlated noise into the latent activations of the policy network to induce a full-covariance action distribution. All learn directly in the full A-dimensional space. Plexus on the other hand, belongs to the category of methods whose action distributions have lower-dimensional support or control, we address these in the next paragraph.

Learning in a latent space. Methods that compress the action space differ in how the decoder is obtained, and each choice carries some requirements summarized in Appendix A. SAR (Berg et al., 2024) discovers synergies by applying ICA and PCA to activations from a policy trained on a simpler task and freezes the result, at the cost of a two-phase pipeline decoupling synergy discovery from the downstream task. DynSyn (He et al., 2024) identifies synergistic muscle groups from the dynamical structure of the system and adapts them within the RL loop; unlike SAR its groups are state-dependent, but a discovery phase is still required. ETL (Wang et al., 2026) fits a Gaussian-mixture variational autoencoder to actions gathered in a dedicated exploration phase and fine-tunes the decoder online by reconstruction. JoSE (Heald et al., 2026) learns a control-affine dynamics model and derives in closed form the state-dependent precoder maximizing the mutual information between latent actions and next joint velocities. Though its decoder is not trained by the RL objective, it is the only prior method to correct its entropy for the volume that decoder induces. λ-hold (Lee et al., 2026) derives its decoder from physiology, commanding per-muscle equilibrium-point threshold lengths from which a stretch-reflex recruitment law computes excitations, and holding each within a gait phase; the map is fixed and state-dependent, and the reduction comes from the reparameterization rather than from a lower-dimensional latent. Ours is a free parameter of the actor trained by reward alone, requiring no extra procedure or preliminary routine.

Accounting for a change in geometry. When the policy acts through a non-invertible decoder, the correct density adjustment is not a Jacobian determinant but the volume element on the image manifold, an object of dimension K (the latent width), familiar from injective and manifold-valued flows (Brehmer & Cranmer, 2020; Caterini et al., 2021; Ross & Cresswell, 2021). Flow-based MaxEnt policies (Haarnoja et al., 2018a; Mazoure et al., 2020; Chao et al., 2024) learn their transformation from reward, but in the square, invertible setting, where the image remains full-dimensional and no direction can be lost. The decoders above, JoSE's aside, take neither route and regularize entropy in the latent space alone. We introduce this method as its own dedicated latent-entropy baseline in Section 4.2. To our knowledge no prior work both learns the decoder from reward and accounts for the geometry it induces, which is the regime in which the volume element doubles as a regularizer on the decoder's Jacobian, preventing rank collapse.

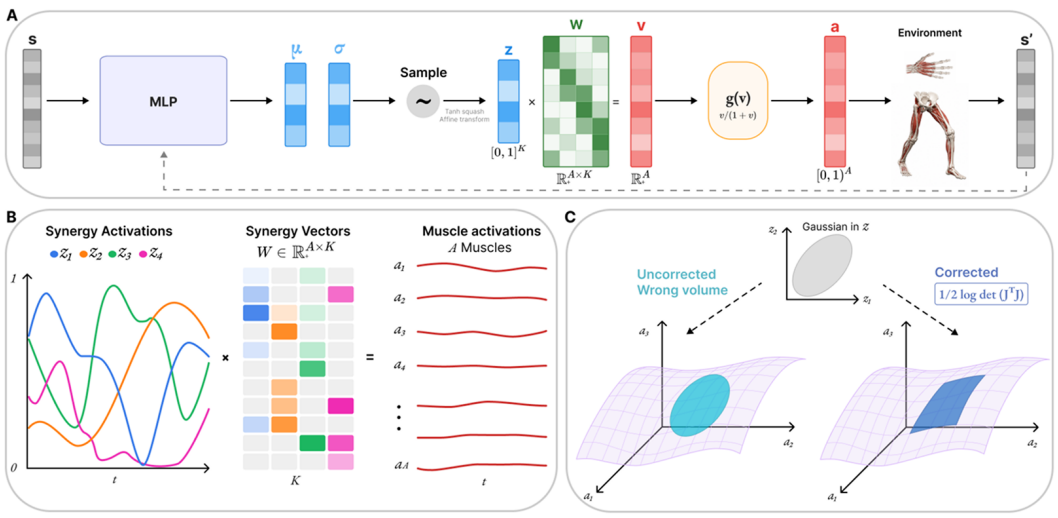

Figure 1: **(A)** Policy architecture. The state s parameterizes a diagonal Gaussian over a K-dimensional latent. The sample is tanh-squashed and affinely shifted to $[0, 1]$, multiplied by the non-negative synergy matrix W to give pre-activations v and elementwise squashed to muscle activations a driving the musculoskeletal environment. **(B)** Along an episode, the decoder factors actions into time-varying synergy activations $z(t)$ and a fixed synergy basis W (middle): each muscle trajectory $a_i(t)$ (right) is a non-negative combination of the corresponding synergies. **(C)** The decoder confines actions to a K-dimensional manifold in $\mathbb{R}^A$. A Gaussian in z (top) pushes forward to a distribution on this manifold. Latent entropy alone measures volume in z-space and gives the wrong on-manifold volume (left, cyan), while adding the Gram-determinant correction to the actor loss (Eq. 11) recovers the geometrically correct on-manifold entropy (right, blue).

4 Method

We call our method **Plexus**, after the anatomical structures through which a small number of neural signals are distributed to many muscles. We describe the architecture in Figure 1.

4.1 Synergy-structured decoder

In motor neuroscience, electromyographic (EMG) recordings are the primary method for measuring muscle activity in experimental settings. Observations show that muscle activation patterns can be well-approximated by a smaller set of basis vectors. Ivanenko et al. (2004) demonstrated that five synergies explain a large fraction of variance during locomotion. Muscle synergies are conventionally estimated by non-negative matrix factorization (NMF) (Lee & Seung, 1999; Tresch et al., 2006): if $M \in \mathbb{R}_+^{A\times T}$ collects A muscle activations over T time steps, one seeks non-negative factors

$$M \approx WC, \qquad W \in \mathbb{R}_+^{A\times K}, \quad C \in \mathbb{R}_+^{K\times T}, \quad K \le A, \tag{4}$$

where the columns of W are *synergy vectors*, fixed muscle weighting patterns, and C holds the time-varying *synergy activations*. Our decoder mirrors Eq. 4, with a latent policy in the role of C and a learnable non-negative matrix in the role of W; the difference is that both are learned from reward alone by interacting with the environment rather than by factorizing recorded data.

Latent policy. A neural network maps the observation s to the sufficient statistics $(\mu_\phi(s), \sigma_\phi(s))$ of a diagonal Gaussian in $\mathbb{R}^K$. A sample $u \sim \mathcal{N}(\mu_\phi(s), \sigma_\phi^2(s))$ is passed through an elementwise $\tanh$ followed by an affine rescaling to $[0, 1]^K$:

$$z = \tfrac{1}{2}\big(\tanh(u) + 1\big) \in [0, 1]^K. \tag{5}$$

The vector z plays the role of the synergy activations $C(:, t)$ in Eq. 4 and is non-negative and bounded as required by the physiological interpretation. We compute the log-density $\log \pi_Z(z|s)$ by squash correction (Eq. 3) together with the affine Jacobian from the $(-1, 1) \to (0, 1)$ rescaling.

Non-negative decoder. The synergy vectors are the columns of a learnable matrix $W_{\text{raw}} \in \mathbb{R}^{A\times K}$ passed through an elementwise softplus, $W = \zeta(W_{\text{raw}}) \in \mathbb{R}_+^{A\times K}$. The pre-activations $v = Wz \in \mathbb{R}_+^A$ are then squashed to bounded muscle activations,

$$a = g(Wz), \qquad g(v) = \frac{v}{1+v} \in [0,1)^A, \tag{6}$$

We adopt this hyperbolic form for its functional properties: g is smooth, satisfies $g(0) = 0$, is bounded above by 1, and its derivative $g'(v) = (1+v)^{-2}$ decays gracefully, preventing saturation-induced gradient vanishing. The form is reminiscent of Hill's classical muscle model (Hill, 1938), but we use it purely as a saturating activation function. The non-negativity of W and z ensures that every muscle receives a non-negative drive, preventing inter-synergy cancellation by construction.

Decoder flexibility. The single-layer linear decoder $a = g(Wz)$ is the simplest instantiation of the synergy model, but the architecture admits natural extensions. First, the number of synergies K is a user-specified parameter, as in NMF, and can range from a small number up to A. Second, muscle group priors can be imposed by masking or tying entries of W_{raw}. In locomotion for instance, bilateral symmetry can be enforced by tying the weights for left and right limbs. Section 5 details how this quarters the free parameters in W.

4.2 Entropy on the induced manifold

Latent entropy. The most direct approach consists in naively ignoring the decoder geometry and regularizing the latent density. From the actor loss laid out in Eq. 2 we have:

$$\mathcal{L}_\pi^{\text{latent}} = \mathbb{E}_{s,z}\big[\alpha \log \pi_Z(z|s) - Q(s, T(z))\big]. \tag{7}$$

This is well defined and computationally simple, but it conflates entropy in z-space with entropy in action space: when $T = g \circ W$ is not volume-preserving, high latent entropy need not correspond to diverse actions. Moreover, ill-conditioning of W is the natural outcome of unconstrained end-to-end learning. Indeed, gradient descent on the Q-value objective alone has no incentive to maintain well-conditioned columns, so columns that produce similar muscle-activation patterns for the visited states tend to become nearly collinear over training. In practice, we observe that the latent entropy bonus even accelerates this collapse: the objective in Eq. 7 rewards high $\mathcal{H}(z)$ irrespective of the action-space volume, so the policy can inflate its latent entropy by spreading mass along directions corresponding to small singular values of W, the directions that contribute least to action diversity.

Change of variables at $K = A$. When $K = A$, the decoder $T : z \mapsto g(Wz)$ is generically a local diffeomorphism and the standard change-of-variables formula relates the two densities,

$$\pi_A(T(z)|s) \;=\; \frac{\pi_Z(z|s)}{|\det J_T(z)|}\,. \tag{8}$$

Replacing $\log \pi_Z$ in the actor loss with $\log \pi_A = \log \pi_Z - \log|\det J_T|$ corrects the volume distortion and restores the correspondence between the entropy bonus and action-space diversity. If W were orthogonal, its contribution to $|\det J_T|$ would be constant and the correction would carry no information about the basis; for a general learned W it is state- and z-dependent, and thus essential.

Intrinsic manifold entropy. When $K < A$ however, the change-of-variables formula (Eq. 8) no longer applies. The image $\mathcal{M} = T([0,1]^K)$ is generically a K-dimensional submanifold of $\mathbb{R}^A$, so the pushforward $T_\# \pi_Z$ is supported on a set of A-dimensional Lebesgue measure zero: $\det J_T$ is not defined (the Jacobian is rectangular), and the action distribution admits no density with respect to the ambient Lebesgue measure. We instead define entropy with respect to the K-dimensional volume measure induced by the embedding. Measured this way, entropy captures the diversity of actions the policy can produce on the manifold it actually occupies. It directly arises as the $\varepsilon \to 0$ limit of the ambient entropy of an ε-thickening of the policy around $\mathcal{M}$ (Brehmer & Cranmer, 2020; Papamakarios et al., 2021), which we show in Appendix B.1.

We obtain a single expression valid for all $K \le A$ via the area formula. For $T : \mathbb{R}^K \to \mathbb{R}^A$ smooth and injective and any measurable f,

$$\int_{\mathbb{R}^K} f(T(z)) \sqrt{\det\big(J_T(z)^\top J_T(z)\big)}\; dz \;=\; \int_{\mathcal{M}} f(x)\; d\nu(x), \tag{9}$$

where $J_T(z) = \partial T/\partial z \in \mathbb{R}^{A\times K}$ and $\det(J_T(z)^\top J_T(z))$ the Gram determinant of the columns of J_T. Identifying densities and taking logarithms gives,

$$\log \pi_{\mathcal{M}}(T(z)|s) = \log \pi_Z(z|s) - \tfrac{1}{2}\log\det\big(J_T(z)^\top J_T(z)\big). \tag{10}$$

When $K = A$ and J_T is invertible, $\det(J_T^\top J_T) = (\det J_T)^2$ and Eq. 10 reduces to the change-of-variables correction of Eq. 8; when $K < A$, the Gram determinant is the squared K-dimensional volume element and the expression remains valid.

Substituting Eq. 10 into the actor loss yields our objective,

$$\boxed{\mathcal{L}_\pi = \mathbb{E}_{s,z}\big[\alpha \log \pi_Z(z|s) - \tfrac{\alpha}{2}\log\det\big(J_T(z)^\top J_T(z)\big) - Q(s, T(z))\big],} \tag{11}$$

where, for $T(z) = g(Wz)$ with $g(v) = v/(1+v)$, the Jacobian is

$$J_T(z) = \operatorname{diag}\big((1+Wz)^{-2}\big)\, W \in \mathbb{R}^{A\times K}, \tag{12}$$

and the Gram matrix is

$$J_T^\top J_T = W^\top \operatorname{diag}\big((1+Wz)^{-4}\big) W \in \mathbb{R}^{K\times K}. \tag{13}$$

We compute $\frac{1}{2}\log\det(J_T^\top J_T)$ via the Cholesky decomposition of the $K \times K$ Gram matrix with a diagonal regularizer εI_K to ensure numerical stability when W is near-singular early in training. The log-determinant is then $\sum_i \log L_{ii}$, where L is the Cholesky factor. We provide a more detailed derivation in Appendix B.2.

4.3 Gradient flow

Because W_{raw} is a parameter of the actor and J_T depends on z, the volume term in Eq. 11 influences training along three separable routes:

1. **Policy.** J_T depends on z through $\operatorname{diag}((1+Wz)^{-2})$, so the term has a non-zero gradient with respect to ϕ and reshapes the latent distribution.
2. **Decoder.** The term is differentiable with respect to W_{raw} and penalizes small singular values of J_T, so gradient descent shapes the decoder to maintain geometric coverage of the action space.
3. **Temperature.** The entropy estimate entering the dual objective for α becomes $-\mathbb{E}[\log \pi_Z - \frac{1}{2}\log\det(J_T^\top J_T)]$, so the term also acts on the temperature trajectory.

We isolate the effect of each route by defining four variants that differ only in which paths are left open. Variant (a) corresponds to the latent entropy of Eq. 7, with the Jacobian correction term absent altogether. In variant (b) the decoder is held fixed inside the correction: the entropy estimate is geometrically correct and the policy is shaped accordingly, and although the critic still trains W through the forward path, W is detached from the entropy term. Variant (c) is a control on the temperature route, repeating (b) with α fixed rather than autotuned, which is what separates a badly conditioned decoder from a temperature controller reacting to one. Variant (d) is our Plexus formulation, Eq. 11, the only one in which the correction acts directly on the decoder. Appendix C.2 sets out which route each variant leaves open, and we report return and decoder conditioning for all four in Section 5.

4.4 Practical algorithm

We integrate the objective into SAC with automatic entropy tuning (Haarnoja et al., 2018c). The critic operates in the full action space and is trained with the standard soft Bellman backup; the actor minimizes Eq. 11 and updates W_{raw} jointly with ϕ under the same optimizer. The target entropy is $\bar{\mathcal{H}} = -K$, matching SAC's $-\dim(\mathcal{A})$ convention applied to the latent dimension, on the view that the policy's stochasticity budget should scale with the number of synergies rather than the ambient muscle count. The decoder is initialized so that Wz is $O(1)$, placing $g(Wz)$ in the responsive region of the nonlinearity: $W_{\text{raw}} \sim \mathcal{N}(\mu_0, 1)$ with $\mu_0 = \log(\exp(1/K) - 1) - 0.5$, giving $\mathbb{E}[\zeta(W_{\text{raw}})] \approx 1/K$. Algorithm 1 in Appendix C.2 describes one gradient step in full.

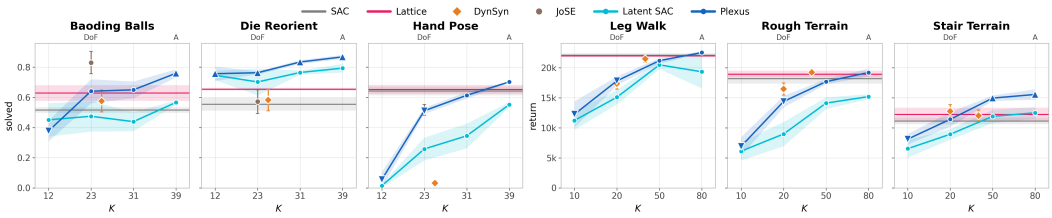

Figure 2: Final score against latent width, mean ± 1 s.e.m. over 5 seeds. Plexus and Latent SAC are shown at every K, while SAC and Lattice are horizontal bands, having no latent space; DynSyn and JoSE are points at their own synergy count. Significance against the strongest full-dimensional baseline (Welch's one-sided t-test at $p < 0.05$) is shown directly on the markers: an up or down triangle where it is above or below and a circle where the test does not separate. Hand tasks are scored on fraction solved, leg tasks on return. Baseline and per-K training curves are in Appendix D.5.

5 Results

Plexus evaluation focuses on (i) overall performance against baselines, and how far the basis can be compressed, (ii) effects of the Jacobian correction on decoder conditioning and performance, and (iii) the interpretability of the architecture and its affordances. We compare Plexus against five baselines, all sharing the SAC training loop, architecture and hyperparameters described in App. C.3.

- **SAC:** (Haarnoja et al., 2018b) Squashed Gaussian on the full A-dimensional action space.
- **Latent SAC:** Plexus' own decoder trained with latent entropy only (Eq. 7).
- **Lattice:** (Chiappa et al., 2023) State-dependent noise injected in policy's latent activations.
- **DynSyn:** (He et al., 2024) Synergistic basis discovered in a preliminary phase and adapted in the RL loop.
- **JoSE:** (Heald et al., 2026) A precoder in closed form from a learned dynamics model, maximising control over joint velocity.

For Plexus and Latent SAC we sweep $K \in \{\text{DoF}/2, \text{DoF}, (A + \text{DoF})/2, A\}$. Each configuration is run with 5 seeds for 25×10^6 environment steps. We use six MyoSuite tasks spanning two models: three hand tasks (Pose Matching, Baoding ball rotation, Die reorientation) (Caggiano et al., 2022b; 2023) on the MyoHand ($A = 39$ muscles, 23 DoF), and three leg tasks (flat walk, rough and stair terrains) on the MyoLeg ($A = 80$ muscles, 20 DoF) (Rajagopal et al., 2016). The hand tasks range from independent joint control to contact-rich dexterous manipulation. The leg tasks test locomotion under increasing terrain difficulty. We provide full environments details in Appendix C.1.

Performance against baselines. In Figure 2, we report final results for all six benchmarks. At $K = A$, Plexus matches or exceeds all full-dimensional baselines on every task, with the largest margins on Die Reorient, Baoding Balls, and Stair Terrain (respectively $+32.7\%$, $+20.7\%$ and $+26.9\%$ against Lattice, all $p < 0.05$) representing contact-rich manipulation and the hardest terrain. Smaller or non-significant gains are seen on Hand Pose, Flat and Rough Terrain walking. Decoder-based baselines behave very differently. DynSyn is consistent: within one s.e.m. of the full-dimensional baselines on all three leg tasks and on both manipulation tasks. It however completely fails to solve Hand Pose, easily explained by the same reason the task does not compress well with Plexus, detailed in the next paragraph. JoSE, as the only baseline also accounting for decoder's geometry, is mixed against us on the hand at $K = \text{DoF}$ (their available highest latent width, since it is a joint-empowerment method): strongest on Baoding (0.830 ± 0.074 against our 0.641 ± 0.080), tied on Hand Pose, and behind on Die Reorient (0.572 ± 0.079 against 0.763 ± 0.009). While balls rotation relies heavily on joint-velocity control, the variable JoSE's precoder maximises, this advantage does not carry to the other two hand tasks. Additionally the method has not been demonstrated on locomotion, where balance and ground contact are not expressible as a function of joint-velocities.

Task compression. We show that how far the basis compresses is a property of the task. Reading the smallest synergy dimension at which Plexus' performance remains comparable to the best full-dimensional baseline, Die Reorient compresses furthest (to $K = \text{DoF}/2$) followed by Baoding and

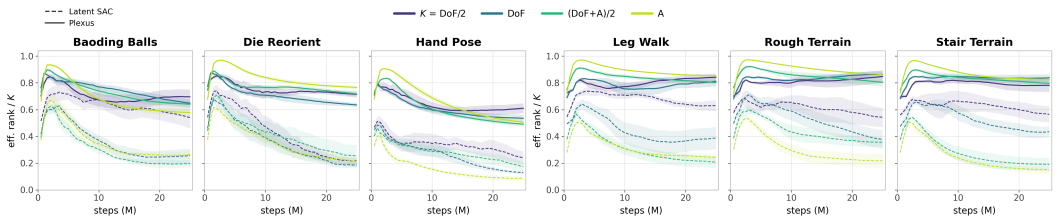

Figure 3: Conditioning of the learned decoder through training: effective rank of W as a fraction of K. *Solid:* Plexus. *Dashed:* Latent SAC. Final ranks against K are in Appendix D.3.

Stair Terrain at $K = \text{DoF}$. Hand Pose and the two easier walking tasks do not compress below $(A + \text{DoF})/2$, for different reasons. Hand Pose requires individuated finger-joint control rather than coordinated groups, so a basis close or narrower than the mechanical degrees of freedom cannot express the target poses at all. Flat and rough walking instead sit near a ceiling every method reaches, which leaves no headroom to absorb a compression cost and makes a small deficit detectable.

Decoder conditioning. Across all tasks and all values of K, Plexus outperforms Latent SAC at every matched latent dimension but one (Baoding at $K{=}12$). Because both methods share the exact same decoder architecture, any gap between them is attributable to the Gram determinant correction alone. To understand why, we directly inspect the geometry of the learned decoder W. Figure 3 follows its effective rank through training. In both Plexus and Latent SAC, the rank rises over the first $\sim$2M steps, then the uncorrected one decays monotonically. As the mechanics in Section 4.2 predict, the decay grows with K since more directions mean more places for the latent entropy bonus to go that cost the Q-value objective nothing. Plexus instead settles at a task-dependent fraction that remains consistent across all given K: between 0.5 and 0.8 on the hand and around 0.8 on the leg.

Gradient path. As we laid out in Section 4.3, variants (a) and (d) bound the spectrum of what is reached by the correction's gradient. Between them, variant (b) lets the term shape the policy and the temperature but cuts its gradient into the decoder, the best available formulation when the decoder is not learned, since variant (d) is not reachable. However, in our case, across the three hand tasks at $K = \text{DoF}$, it is pathological on 13 of 15 seeds. The temperature explodes, with $\log\det(J^\top J)$ driven strongly negative and the Gram matrix turning numerically singular, the correction can no longer be computed and the policy freezes. The loop is self-feeding: as the basis degenerates the entropy estimate falls, the entropy estimate drops below target and α rises. This failure mode needs slack to develop. On Leg Walk at $K = \text{DoF}$, where the basis has barely more directions than the task needs, (b) stays stable; widening the latent to $(A + \text{DoF})/2$ collapses the same variant. Thus, the entropy bonus has somewhere cheap to go only once the decoder holds enough directions to spare.

Variant (c) cuts the gradient into the temperature instead, pinning α at 0.05, the mean of the auto-tuned values over all runs that do not diverge, removing the dual without displacing the temperature from where it would have settled. Every divergence disappears and $\log\det$ returns above zero but rank does not recover fully: only to 0.30 & 0.25 on Baoding at $K = \text{DoF}$ and $(A + \text{DoF})/2$ respectively versus 0.61 & 0.63 for variant (d) and to 0.71 & 0.30 versus 0.83 & 0.78 on Leg Walk for the same latent widths. Consequently, what remains once the runaway is suppressed is the conditioning itself, and only (d) repairs that. The full table with performances is available in Appendix D.6.

Biomechanical priors. A basis discovered elsewhere can be supplied as the initial state of our decoder. Starting W from DynSyn's grouping and letting the reward move it stays within statistical noise of the random initialization on all tasks except Leg Walk, where at matched $K = 20$ it achieves better results (21,311 against 17,813; Appendix D.4). So a discovered basis is worth no harm adopting where one exists but is not a requirement. However, freezing that same basis throughout training fails: hand tasks solved fraction stands at 0 for Hand Pose, 0.22 for Baoding and 0.62 for Die Reorient while locomotion tasks never reach a return above 327, all well below both random and DynSyn initializations. Lee et al. (2026) report the same with an NMF basis distilled from their own best rollouts, which also fails to learn. A basis good enough to reconstruct a solution is thereby not sufficient to find one.

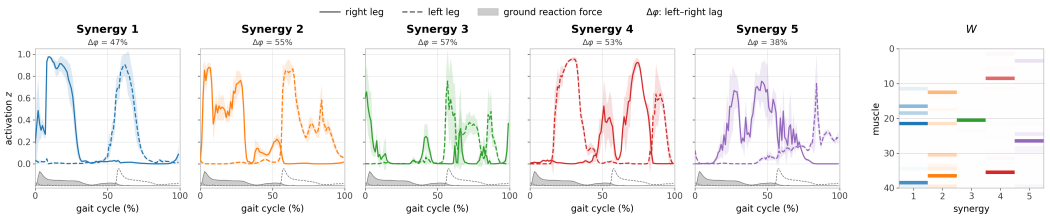

Figure 4: MyoLegWalk with a single non-negative 40×5 decoder shared by both legs, best performing seed. *Panels 1–5:* each synergy's activation over the gait cycle, mean ± 1 s.d. over nine strides with mean ground-reaction force. *Right:* the shared basis W, columns scaled to their peak (actuator names in Table 2). $\Delta\varphi$ gives the left–right phase lag, 50% for a symmetric gait.

Because W is one matrix and not a state-dependent map, we can directly impose structure on it. Tying the two legs as one 40×5 block shared across both action groups, each keeping its own slice of z so the legs remain independently driven, quarters the number of free parameters against an unconstrained 80×10 decoder. In addition, the symmetry greatly improves performance, reaching a mean return of $16.0k \pm 1.1k$, well above the unconstrained decoder at the same ten dimensions ($12.3k \pm 2.2k$). Additionally, the learned synergies can be further inspected to assess their biomechanical plausibility. Figure 4 shows the five synergies discovered by the shared decoder of the best performing seed, with their activation over the gait cycle. Decoder columns are sparse ($1-6$ muscles above 10% of the column's maximum), so each one reads as a muscle group and the corresponding synergy panel says when that group is used. Synergy 1 groups knee extensors with hip abductors (glutes and quadriceps) and peaks just after foot strike, textbook load acceptance and pelvis stabilization. Synergy 4 pairs ankle and toe dorsiflexors and peaks in mid-swing, when the foot has to clear the ground. Four of the five left–right lags fall within 7% of the half-cycle a symmetric gait implies, while the fifth one sits below at 38%. Bilateral symmetry is one prior among many: any structure expressible as a constraint on W can be imposed, and its effect inspected, the same way.

6 Discussion and Limitations

Summary. Plexus makes the synergy basis a free parameter of the actor and learns it from reward alone: no discovery phase, no dynamics model and no reference data. Acting through a learned decoder, however, moves the entropy bonus off the space it should regularize: the cheapest directions for stochasticity are the ones the decoder is not using, so the basis loses rank until the policy steers only a subspace of the latent space. Measuring entropy as volume on the manifold the policy occupies removes that incentive. But shaping the policy and the temperature while holding the decoder out of the correction term is unstable, and pinning the temperature stops the divergence without recovering the rank. Only the route into W does both. Plexus learns one non-negative matrix holding most of its directions, matching or exceeding every full-dimensional baseline at $K = A$ and compressing to the mechanical degrees of freedom or below where the task admits it. The decoder can be further inspected, initialized or constrained with biomechanical priors such as a discovered grouping or symmetry, leading to more interpretable and better-performing policies.

Limitations. By design, because the correction keeps pressure on all K directions and never exposes an unused synergy, the latent width K remains a hyperparameter, as in all other methods cited in this work. The basis is also task-specific, and we make no transfer claim. The correction is a change to the log-density costing one Cholesky factorisation ($\mathcal{O}(K^3)$), negligible here (Appendix F) but costly at larger K. While any latent policy should correct the entropy for the decoder volume, a learned decoder asks for a path from reward to W. SAC supplies it through the reparameterised actor but an on-policy surrogate does not, since its importance ratio never touches W, so such instantiations would need an added pathwise term. Finally, learning from reward alone also means the reward decides what is learned, and a return is a weak specification: several gait styles reach the same score, so neither the behaviour nor the basis is pinned down. With MuJoCo's rigid tendons, fixed pennation angle and contact handling increasing the sim-to-real gap on top, we read the synergies as a characterisation only, not validation. Nonetheless, we expect the functional claims to carry to a higher-fidelity simulator such as Hyfydy or OpenSim more readily.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing parts of the method in code, to provide feedback on and discuss the experimental design, and to interpret and visualize results. We did not use generative AI tools to generate synthetic datasets, develop the theoretical framework, propose or refine hypotheses, formulate mathematical claims, contribute to any part of the derivations, or assist in writing any proofs. Data translation, dataset cleaning and reformatting, and qualitative or thematic data analysis were not applicable to this work. We additionally used generative AI tools to edit the manuscript for readability and conciseness, check the consistency of notation and cross-references, write and refactor analysis and plotting code, generate figures and tables from logged experimental runs, search for and summarize related work, and format references. All AI-assisted work was reviewed by the authors. Every sentence was reviewed and revised by the authors; every citation retained in the manuscript was verified against its original source; and all AI-assisted code was verified by the authors. All reported results and numerical values come from experiments executed by the authors using the code provided in the supplementary material. The authors take full responsibility for the final content of this work, including all text, claims, analyses, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work involves no human subjects, no human or animal data, and no personally identifiable information. All experiments are run in simulation on the publicly available MyoSuite benchmark and its MyoHand and MyoLeg models, used under their respective licences. The ethical consideration specific to this work is interpretive. The synergy basis we obtain should not be carried into rehabilitation, prosthetics or exoskeleton control without validation. We see no dual-use concern beyond those general to reinforcement learning for control.

REPRODUCIBILITY STATEMENT

We have taken care to make every result in this paper reproducible. The method section and appendix specify the architecture, the objective and the gradient path of each variant, and include the derivation of the decoder's Jacobian and Gram matrix, pseudocode for one training step, and the complete set of hyperparameters for our method and every baseline. The supplementary material contains the full training code, in which all methods share a single codebase, together with one configuration file per method, the precomputed muscle groups used by the DynSyn baseline, and a lock file pinning every dependency version. All experiments use the publicly available MyoSuite benchmark, and the compute used is reported in the appendix.

REFERENCES

Cameron Berg, Vittorio Caggiano, and Vikash Kumar. SAR: generalization of physiological agility and dexterity via synergistic action representation. *Autonomous Robots*, 48(8), November 2024. doi: 10.1007/s10514-024-10182-4.

Nikolai A. Bernstein. *The Co-ordination and Regulation of Movements*. Pergamon Press, Oxford, 1967.

Emilio Bizzi, Vincent C. K. Cheung, Andrea d'Avella, Philippe Saltiel, and Matthew Tresch. Combining modules for movement. *Brain Research Reviews*, 57(1):125–133, 2008. ISSN 0165-0173. doi: 10.1016/j.brainresrev.2007.08.004. Networks in Motion.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 442–453, 2020.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. MyoSuite: A contact-rich simulation suite for musculoskeletal motor control. In *Learning for Dynamics and Control Conference (L4DC)*, volume 168 of *Proceedings of Machine Learning Research*, pp. 492–507. PMLR, 2022a.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Seungmoon Song, Yuval Tassa, Massimo Sartori, and Vikash Kumar. MyoChallenge 2022: Learning contact-rich manipulation using a musculoskeletal hand. NeurIPS 2022 Competition Track. https://sites.google.com/view/myosuite/myochallenge/myochallenge-2022, 2022b.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Seungmoon Song, Chun Kwang Tan, Cameron Berg, Pierre Schumacher, Massimo Sartori, and Vikash Kumar. MyoChallenge 2023: Towards human-level dexterity and agility. NeurIPS 2023 Competition Track. https://sites.google.com/view/myosuite/myochallenge/myochallenge-2023, 2023.

Anthony L. Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P. Cunningham. Rectangular flows for manifold learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Chen-Hao Chao, Chien Feng, Wei-Fang Sun, Cheng-Kuang Lee, Simon See, and Chun-Yi Lee. Maximum entropy reinforcement learning via energy-based normalizing flow. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Alberto Silvio Chiappa, Alessandro Marin Vargas, Ann Zixiang Huang, and Alexander Mathis. Latent exploration for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Alberto Silvio Chiappa, Pablo Tano, Nisheet Patel, Abigaïl Ingster, Alexandre Pouget, and Alexander Mathis. Acquiring musculoskeletal skills with curriculum-based reinforcement learning. *Neuron*, 112(23):3969–3983.e5, December 2024. doi: 10.1016/j.neuron.2024.09.002.

Andrea d'Avella, Philippe Saltiel, and Emilio Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6(3):300–308, March 2003. doi: 10.1038/nn1010.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=PtSAD3caaA2.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1851–1860. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018b.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018c. doi: 10.48550/arXiv.1812.05905.

Kaibo He, Chenhui Zuo, Chengtian Ma, and Yanan Sui. DynSyn: Dynamical synergistic representation for efficient learning and control in overactuated embodied systems. In *International Conference on Machine Learning (ICML)*, volume 235, pp. 18115–18132, 2024.

James Heald, Vittorio Caggiano, Vikash Kumar, and Maneesh Sahani. Joint-space empowerment for dexterous coordination in tendon-driven hands. In *International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=qI2eHwfNfh.

Archibald V. Hill. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 126(843):136–195, 1938. doi: 10.1098/rspb.1938.0050.

Yuri P. Ivanenko, Richard E. Poppele, and Francesco Lacquaniti. Five basic muscle activation patterns account for muscle activity during human locomotion. *The Journal of Physiology*, 556(1): 267–282, 2004. doi: 10.1113/jphysiol.2003.057174.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. doi: 10.1038/44565.

Jun Hyuk Lee, Chihyeong Lee, and Jooeun Ahn. Lambda-hold control: Human-like movement emerges from a minimal task reward in predictive musculoskeletal simulation. *arXiv preprint arXiv:2608.17030*, 2026. doi: 10.48550/arXiv.2608.17030. URL https://arxiv.org/abs/2608.17030.

Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*, pp. 430–444. PMLR, 2020.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL https://jmlr.org/papers/v22/19-1028.html.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1634–1644. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/raffin22a.html.

Apoorva Rajagopal, Christopher L. Dembia, Matthew S. DeMers, Denny D. Delp, Jennifer L. Hicks, and Scott L. Delp. Full-body musculoskeletal model for muscle-driven simulation of human gait. *IEEE Transactions on Biomedical Engineering*, 63(10):2068–2079, October 2016. doi: 10.1109/TBME.2016.2586891.

Brendan Ross and Jesse Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26635–26648. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/dfd786998e082758be12670d856df755-Paper.pdf.

Pierre Schumacher, Daniel F.B. Haeufle, Dieter Büchler, Syn Schmitt, and Georg Martius. DEP-RL: Embodied exploration for reinforcement learning in overactuated and musculoskeletal systems. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=C-xa_D3oTj6.

Pierre Schumacher, Thomas Geijtenbeek, Vittorio Caggiano, Vikash Kumar, Syn Schmitt, Georg Martius, and Daniel F.B. Haeufle. Emergence of natural and robust bipedal walking by learning from biologically plausible objectives. *iScience*, 28(4):112203, 2025. ISSN 2589-0042. doi: 10.1016/j.isci.2025.112203.

Ajay Seth, Jennifer L. Hicks, Thomas K. Uchida, Ayman Habib, Christopher L. Dembia, James J. Dunne, Carmichael F. Ong, Matthew S. DeMers, Apoorva Rajagopal, Matthew Millard, Samuel R. Hamner, Edith M. Arnold, Jennifer R. Yong, Shrinidhi K. Lakshmikanth, Michael A. Sherman, Joy P. Ku, and Scott L. Delp. OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLOS Computational Biology*, 14 (7):e1006223, 2018. doi: 10.1371/journal.pcbi.1006223.

Merkourios Simos, Alberto Silvio Chiappa, and Alexander Mathis. Kinesis: Motion imitation for human musculoskeletal locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. doi: 10.48550/arXiv.2503.14637. URL https://arxiv.org/abs/2503.14637.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Matthew C. Tresch, Philippe Saltiel, and Emilio Bizzi. The construction of movement by the spinal cord. *Nature Neuroscience*, 2(2):162–167, 1999. doi: 10.1038/5721.

Matthew C. Tresch, Vincent C. K. Cheung, and Andrea d'Avella. Matrix factorization algorithms for the identification of muscle synergies: Evaluation on simulated and experimental data sets. *Journal of Neurophysiology*, 95(4):2199–2212, 2006. doi: 10.1152/jn.00222.2005.

Huawei Wang, Vittorio Caggiano, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. MyoSim: Fast and physiologically realistic MuJoCo models for musculoskeletal and exoskeletal studies. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8104–8111, 2022. doi: 10.1109/ICRA46639.2022.9811684.

Yiming Wang, Kaiyan Zhao, Xu Li, Yan Li, Jiayu Chen, Steven Morad, and Leong Hou U. Explore to learn: Latent exploration through disentangled synergy patterns for reinforcement learning in overactuated control. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 26670–26678, 2026. doi: 10.1609/aaai.v40i31.39876. URL `https://ojs.aaai.org/index.php/AAAI/article/view/39876`.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1433–1438, 2008.

A ACTION DECODERS IN PRIOR WORK

Table 1 expands Section 3 summarizing where each method's decoder comes from, what obtaining it costs beyond the reward, and whether the method accounts for the volume of the manifold that decoder induces.

Table 1: Where the action decoder comes from and what obtaining it requires. The last column indicates whether the method accounts for the volume of the decoder manifold.

Method	**Decoder origin**	**Requires beyond reward**	**Correction**
SAR	ICA/PCA on pre-training activations	a pre-training policy and rollouts	no
DynSyn	clustering by dynamical similarity	a muscle similarity measure	no
ETL	GMVAE on explored actions	exploration phase, reconstruction target	no
λ-hold	equilibrium-point reparameterization	a muscle model and a reflex law	no
JoSE	closed form from learned dynamics	dynamics model and controlled variable	yes
Latent SAC	free parameter of the actor	—	no
Ours	free parameter of the actor	—	yes

B THEORY

B.1 INTRINSIC ENTROPY VIA ε-THICKENING

This section shows that the intrinsic entropy in Sec. 4.2 is the policy-relevant part of a well-defined ambient entropy, in the limit of vanishing noise. Because the action $X = T(Z)$ lies on the K-dimensional manifold $\mathcal{M} \subset \mathbb{R}^A$, it has no density in $\mathbb{R}^A$ and $h(X \mid s)$ is ill-defined. Adding small independent isotropic Gaussian noise regularises this:

$$Y \;:=\; T(Z) + \sigma\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I_A), \qquad \sigma > 0, \tag{14}$$

so that $Y \mid s$ has a density and $h(Y \mid s)$ is well-defined.

Orthogonal frame. Let $U(z) \in \mathbb{R}^{A\times K}$ have orthonormal columns in the directions $\mathcal{M}$ moves at $T(z)$ (i.e. $\mathrm{Im}(J_T(z))$), and $V(z) \in \mathbb{R}^{A\times(A-K)}$ span the remaining $A-K$ orthogonal directions. Define $B(z) := U(z)^\top J_T(z) \in \mathbb{R}^{K\times K}$; then $J_T(z) = U(z)B(z)$, $B(z)$ is invertible, and

$$J_T(z)^\top J_T(z) = B(z)^\top B(z), \qquad \log|\det B(z)| = \tfrac{1}{2}\log\det\big(J_T(z)^\top J_T(z)\big). \tag{15}$$

Tangent–normal split. Project Y onto the two parts of the frame at Z:

$$Y_\parallel \;:=\; U(Z)^\top Y \;\in\; \mathbb{R}^K, \qquad Y_\perp \;:=\; V(Z)^\top\big(Y - T(Z)\big) \;\in\; \mathbb{R}^{A-K}. \tag{16}$$

Given $Z = z, Y - T(z) = \sigma\varepsilon$, and projecting an isotropic Gaussian onto orthogonal subspaces gives independent Gaussians, so $Y_\perp \mid Z = z \sim \mathcal{N}(0, \sigma^2 I_{A-K})$ for every z. Thus $Y_\perp$ does not depend on Z and carries no information about the policy; its entropy is the explicit formula for a Gaussian,

$$h(Y_\perp) \;=\; \tfrac{A-K}{2}\log(2\pi e\sigma^2). \tag{17}$$

Since $Y_\parallel$ and $Y_\perp$ are conditionally independent given Z, and $(Y_\parallel, Y_\perp)$ determines Y given Z,

$$h(Y \mid s) \;=\; h(Y_\parallel \mid s) \;+\; h(Y_\perp). \tag{18}$$

Small-noise limit. As $\sigma \to 0$, $h(Y_\parallel \mid s)$ converges to the intrinsic entropy $h_{\mathcal{M}}(X \mid s)$, which by the area formula equation 9 and equation 15 equals

$$h_{\mathcal{M}}(X \mid s) \;=\; h(Z \mid s) + \mathbb{E}_{z\sim\pi_Z}\big[\tfrac{1}{2}\log\det(J_T(z)^\top J_T(z))\big], \tag{19}$$

matching Eq. 10. Putting it together,

$$h(Y \mid s) \;=\; h_{\mathcal{M}}(X \mid s) \;+\; \tfrac{A-K}{2}\log(2\pi e\sigma^2) \;+\; o(1) \qquad (\sigma \to 0). \tag{20}$$

The second term diverges but is independent of the policy and decoder, so it drops out of the actor gradient. What remains is exactly equation 10.

B.2 Jacobian and Gram matrix of the Plexus decoder

We derive Eqs. equation 12–equation 13 from the chain rule. The decoder factors as $T = g \circ \ell$ where $\ell(z) = Wz$ and $g(v)_i = v_i/(1+v_i)$; the upstream squash $\psi(u) = \frac{1}{2}(\tanh u + 1)$ is handled by the standard correction equation 3 and does not enter J_T.

Jacobian. Since g is elementwise, $\partial g/\partial v = \mathrm{diag}((1+v)^{-2})$, and the chain rule gives

$$J_T(z) \;=\; \mathrm{diag}\big((1+Wz)^{-2}\big)\, W \;\in\; \mathbb{R}^{A\times K}, \tag{21}$$

reproducing Eq. 12. The diagonal is full rank for any finite Wz, so $\mathrm{rank}(J_T) = \mathrm{rank}(W)$.

Gram matrix. Writing $D = \mathrm{diag}((1+Wz)^{-2})$ and using $D^\top D = D^2$ recovers Eq. 13:

$$J_T^\top J_T \;=\; W^\top \mathrm{diag}\big((1+Wz)^{-4}\big)\, W \;\in\; \mathbb{R}^{K\times K}. \tag{22}$$

At $K = A$, $\det(J_T^\top J_T) = (\det J_T)^2$, recovering the standard change-of-variables formula. At $K < A$, J_T is rectangular and $\det J_T$ is undefined; $\sqrt{\det(J_T^\top J_T)}$ measures the K-dimensional volume spanned by the columns of J_T, which is exactly the weight needed in the area formula equation 9.

Numerical computation. We form the $K \times K$ Gram $G = W^\top \mathrm{diag}((1+Wz)^{-4})W + \varepsilon I_K$ directly and compute $\frac{1}{2}\log\det G = \sum_i \log L_{ii}$ via Cholesky. The εI_K term guards against ill-conditioning early in training; Appendix E.2 shows performance is insensitive to ε over several orders of magnitude.

C Implementation details

C.1 Environments

We evaluate Plexus on six MyoSuite (Caggiano et al., 2022a) tasks over two musculoskeletal models, spanning static posing, contact-rich object manipulation, and locomotion over three terrains.

- **Hand Pose**: The MyoHand model ($A = 39$ muscles, 23 DoF) has to match random target poses. This is the simplest task in our evaluation and serves partly as a sanity check: reaching arbitrary joint configurations requires independent control of individual DoF, a setting where low-dimensional synergy representations are not expected to help.
- **Baoding ball rotation**: The same MyoHand model controls two balls that must be continuously rotated around each other in the palm. This task was introduced as part of the NeurIPS 2022 MyoChallenge (Caggiano et al., 2022b) and demands temporally coordinated, fine-grained dexterous manipulation.
- **Die reorientation**: The MyoHand model must reorient a die to a target face, also from the NeurIPS 2022 MyoChallenge (Caggiano et al., 2022b). This tests precise multi-finger coordination with contact-rich dynamics.
- **Leg walk**: The MyoLeg model ($A = 80$ muscles, 20 DoF) is derived from the Rajagopal musculoskeletal model (Rajagopal et al., 2016) and serves as the locomotion platform for the MyoChallenge chase-tag task (Caggiano et al., 2023). It has to walk straight on flat terrain.
- **Rough or Stair terrain walk**: The same MyoLeg model must walk across rough irregular terrain or up a staircase.

Note that for the hand-object interaction environments, we use the implementation made available by Chiappa et al. (2023) where they add an "alive" reward to prevent early termination by dropping the object.

Table 2 lists all muscles composing the MyoHand and MyoLeg musculoskeletal models.

Table 2: Musculoskeletal models. The index column matches the row ordering in the activation vector a and hence in the synergy vectors and muscle activations panels of the case-study figures (Fig. 9 and 10).

(a) MyoHand: 39 muscle-tendon units, 23 DoF, 29 bones. Indices 0–38 map directly to the 39-dimensional action vector.

Idx	Full name
0	Extensor Carpi Radialis Longus
1	Extensor Carpi Radialis Brevis
2	Extensor Carpi Ulnaris
3	Flexor Carpi Radialis
4	Flexor Carpi Ulnaris
5	Palmaris Longus
6	Pronator Teres
7	Pronator Quadratus
8	Extensor Indicis Proprius
9	Extensor Pollicis Longus
10	Extensor Pollicis Brevis
11	Flexor Pollicis Longus
12	Abductor Pollicis Longus
13	Opponens Pollicis
14	Flexor Digitorum Superficialis 2 (index)
15	Flexor Digitorum Superficialis 3 (middle)
16	Flexor Digitorum Superficialis 4 (ring)
17	Flexor Digitorum Superficialis 5 (little)
18	Flexor Digitorum Profundus 2 (index)
19	Flexor Digitorum Profundus 3 (middle)
20	Flexor Digitorum Profundus 4 (ring)
21	Flexor Digitorum Profundus 5 (little)
22	Extensor Digitorum Communis 2 (index)
23	Extensor Digitorum Communis 3 (middle)
24	Extensor Digitorum Communis 4 (ring)
25	Extensor Digitorum Communis 5 (little)
26	Extensor Digiti Minimi
27	Radial Interosseous 2 (index)
28	Radial Interosseous 3 (middle)
29	Radial Interosseous 4 (ring)
30	Radial Interosseous 5 (little)
31	Lumbrical 2 (index)
32	Lumbrical 3 (middle)
33	Lumbrical 4 (ring)
34	Lumbrical 5 (little)
35	Palmar or Ulnar Interosseous 2 (index)
36	Palmar or Ulnar Interosseous 3 (middle)
37	Palmar or Ulnar Interosseous 4 (ring)
38	Palmar or Ulnar Interosseous 5 (little)

(b) MyoLeg: 80 muscle-tendon units, 20 DoF, 10 joints. Indices 0–39 are right-side muscles; indices 40–79 are their symmetric left-side counterparts.

Idx	Full name
0	Adductor brevis
1	Adductor longus
2	Adductor magnus (distal)
3	Adductor magnus (ischial)
4	Adductor magnus (middle)
5	Adductor magnus (proximal)
6	Biceps femoris long head
7	Biceps femoris short head
8	Extensor digitorum longus
9	Extensor hallucis longus
10	Flexor digitorum longus
11	Flexor hallucis longus
12	Gastrocnemius lateral head
13	Gastrocnemius medial head
14	Gluteus maximus (superior)
15	Gluteus maximus (middle)
16	Gluteus maximus (inferior)
17	Gluteus medius (anterior)
18	Gluteus medius (middle)
19	Gluteus medius (posterior)
20	Gluteus minimus (anterior)
21	Gluteus minimus (middle)
22	Gluteus minimus (posterior)
23	Gracilis
24	Iliacus
25	Peroneus brevis
26	Peroneus longus
27	Piriformis
28	Psoas
29	Rectus femoris
30	Sartorius
31	Semimembranosus
32	Semitendinosus
33	Soleus
34	Tensor fascia latae
35	Tibialis anterior
36	Tibialis posterior
37	Vastus intermedius
38	Vastus lateralis
39	Vastus medialis

C.2 Algorithm and variants

Algorithm 1 states one gradient step of the full method, variant (d) of Table 3. The other variants are small edits to it: (a) drops the $\log \mathcal{J}$ term from line 11, and (b) recomputes that term with W_{raw} detached inside the Gram matrix while leaving W attached in the decoder path, so the critic still trains the decoder through $Q(s, T(z))$. Variant (c) is (b) with the temperature update disabled and α pinned at 0.05, leaving line 11 as in (b). Note that (d) is undefined whenever the decoder is not a parameter of the actor.

Table 3: Variants separating the roles of the volume term.

	Variant	Policy	Decoder	Temperature
(a)	Latent SAC: term omitted (Eq. 7)	—	—	—
(b)	Stop-gradient on W_{raw}	✓	—	✓
(c)	(b) with α fixed	✓	—	Fixed
(d)	Plexus: full (Eq. 11)	✓	✓	✓

Algorithm 1 One gradient step.

1: Sample batch $(s, a, r, s') \sim \mathcal{D}$
2: **Critic update:** standard SAC soft Bellman backup
3: **Actor update:**
4: Sample $u \sim \mathcal{N}(\mu_\phi(s), \sigma^2_\phi(s))$
5: $z \leftarrow \frac{1}{2}(\tanh(u) + 1)$
6: $W \leftarrow \zeta(W_{\text{raw}})$
7: $a \leftarrow g(Wz), \quad g(v)_i = v_i/(1 + v_i)$
8: Compute $\log \pi_Z(z|s)$ via squash + affine correction
9: $G \leftarrow W^\top \operatorname{diag}\big((1 + Wz)^{-4}\big)W + \varepsilon I_K$
10: $L \leftarrow \text{Cholesky}(G); \quad \log \mathcal{J} \leftarrow \sum_i \log L_{ii}$ $\{= \frac{1}{2} \log \det(J_T^\top J_T)\}$
11: $\mathcal{L}_\pi \leftarrow \alpha \log \pi_Z(z|s) - \alpha \log \mathcal{J} - Q(s, a)$
12: Update ϕ, W_{raw} to minimize $\mathcal{L}_\pi$
13: **Temperature update:** dual gradient descent on α with target entropy $\bar{\mathcal{H}} = -K$

C.3 Hyperparameters

Table 4 reports the full set of hyperparameters used for Plexus and all baselines. All methods share the same SAC training loop, critic architecture, optimizer, replay buffer size, batch size, and learning rates unless explicitly noted; method-specific hyperparameters are listed in their own row. Numbers reported in this paper are therefore not to be carried across implementations, and every comparison is made inside a single codebase. This ensures a fair evaluation of each method and can explain why results may differ from those reported in the original papers; where they differ, our numbers are usually the higher of the two.

Table 4: Complete hyperparameter list. Shared settings apply to every method; method-specific blocks list deviations or additions.

Hyperparameter	**Value**
Optimisation (shared)	
Optimizer	Adam
Actor learning rate	10^{-4}
Critic learning rate	3×10^{-4}
Temperature (α) learning rate	10^{-3}
Gradient clip norm	20
Off-policy training (shared)	
Batch size	256
Discount γ	0.99
Polyak τ	5×10^{-3}
Replay buffer capacity	10^6
Learning starts	25 000 steps
n-step returns	3
Total environment steps	25×10^6
Parallel environments	30
Network architecture (shared)	
Hidden layers $\times$ units	2×256 (actor and critic)
Activation	SiLU
Layer normalization	yes
Weight initialization	orthogonal
Observation normalization	running mean/std
SAC temperature (shared)	
α initialization	1.0
Automatic entropy tuning	yes
Target entropy (full-dim SAC)	$-A$
Plexus and Latent SAC	
K sweep (MyoHand, A=39, DoF=23)	$\{12,\ 23,\ 31,\ 39\}$
K sweep (MyoLeg, A=80, DoF=20)	$\{10,\ 20,\ 50,\ 80\}$
Non-negativity	$\zeta(W_{\text{raw}})$
Final squash g	$v/(1+v)$
W_{raw} init shift μ_0	$\log(\exp(1/K)-1)-0.5$
Target entropy $\bar{\mathcal{H}}$	$-K$
Cholesky regularizer ε (Plexus only)	10^{-4}
Lattice (Chiappa et al., 2023)	
Re-sampling frequency	1
Std clip	$[10^{-3},\ 1.0]$
Std regularizer	10^{-4}
Mean clip	2.0
DynSyn (He et al., 2024)	
Synergy groups K (MyoHand / MyoLeg)	25 / (20, 40)
Ramp onset	10^6 steps
Ramp slope	5×10^{-8}
Max amplitude deviation	$\pm 10\%$
Target entropy $\bar{\mathcal{H}}$	$-K$
JoSE (Heald et al., 2026)	
Latent dimension K used	23 ($=$ DoF $=\dim \mathcal{Y}$, full-rank case), MyoHand only
Controlled variable y	joint angular velocities
Dynamics network	hidden $[400, 300]$, lr 10^{-5}, MC dropout 0.5
Precoder	row-normalised basis of $\mathrm{row}(G(s))$, per state

D ADDITIONAL RESULTS

D.1 FULL COMPARISON TABLE

Tables 5 and 6 give results for every method at every latent width it was run at, complementary to each operating point in Figure 2. MyoHand is scored on fraction solved and MyoLeg on return, results show the mean $\pm$ 1 s.e.m. over 5 seeds, each seed scored on the three evaluations directly following the 25×10^6 steps budget. Best arm per column in bold, with * where it exceeds the best arm of every other method under a one-sided Welch test at $p < 0.05$.

Table 5: MyoHand.

Method	K	Baoding Balls	Die Reorient	Hand Pose
Full-dimensional				
SAC	39	0.516 $\pm$0.018	0.555 $\pm$0.043	0.643 $\pm$0.029
Lattice	39	0.629 $\pm$0.052	0.654 $\pm$0.009	0.653 $\pm$0.030
Decoder-based				
DynSyn	25	0.575 $\pm$0.072	0.582 $\pm$0.071	0.033 $\pm$0.011
JoSE	23	**0.830** $\pm$0.074	0.572 $\pm$0.079	0.517 $\pm$0.036
Ours				
Latent SAC	12	0.451 $\pm$0.110	0.746 $\pm$0.031	0.014 $\pm$0.011
	23	0.475 $\pm$0.101	0.702 $\pm$0.088	0.258 $\pm$0.075
	31	0.439 $\pm$0.063	0.765 $\pm$0.015	0.345 $\pm$0.081
	39	0.566 $\pm$0.014	0.793 $\pm$0.028	0.552 $\pm$0.017
Plexus	12	0.378 $\pm$0.068	0.757 $\pm$0.047	0.056 $\pm$0.051
	23	0.641 $\pm$0.080	0.763 $\pm$0.009	0.511 $\pm$0.020
	31	0.650 $\pm$0.055	0.834 $\pm$0.013	0.614 $\pm$0.022
	39	0.759 $\pm$0.027	**0.868** $\pm$0.013*	**0.702** $\pm$0.004

Table 6: MyoLeg.

Method	K	Leg Walk	Rough Terrain	Stair Terrain
Full-dimensional				
SAC	80	22,071 $\pm$315	18,211 $\pm$845	11,147 $\pm$546
Lattice	80	21,997 $\pm$140	18,942 $\pm$581	12,253 $\pm$1,078
Decoder-based				
DynSyn	20	17,357 $\pm$922	16,457 $\pm$1,018	12,740 $\pm$1,122
	40	21,450 $\pm$583	**19,240** $\pm$314	12,022 $\pm$958
Ours				
Latent SAC	10	11,224 $\pm$1,522	6,119 $\pm$1,510	6,533 $\pm$1,465
	20	15,099 $\pm$1,179	8,973 $\pm$2,017	8,934 $\pm$867
	50	20,475 $\pm$696	14,110 $\pm$963	11,909 $\pm$1,095
	80	19,320 $\pm$2,729	15,190 $\pm$419	12,500 $\pm$967
Plexus	10	12,265 $\pm$2,151	6,923 $\pm$1,559	8,121 $\pm$862
	20	17,813 $\pm$732	14,378 $\pm$947	11,413 $\pm$1,212
	50	21,169 $\pm$457	17,677 $\pm$436	14,944 $\pm$457
	80	**22,524** $\pm$188	19,202 $\pm$637	**15,551** $\pm$878*

D.2 Effect of the volume term

Table 7 gives results for Plexus against the identical decoder trained without the volume term, at matched K, as a relative difference in score. All 24 cells except Baoding at the narrowest K value are positive. And 16 of them, shown in bold, exceed the combined s.e.m.

Table 7: The volume term effect on all 24 paired cells. Δ is the relative difference in score between Plexus and the identical decoder trained without the term at matched K, positive favours the term. Bold marks the cells whose margin exceeds the combined s.e.m. The last column averages Δ over the four widths of each task.

Task	K	Δ	mean
Baoding Balls	DoF/2	-16.3%	*+25.2%*
	DoF	**+34.9%**	
	$(A + \text{DoF})/2$	**+48.0%**	
	A	**+34.1%**	
Die Reorient	DoF/2	+1.4%	*+7.1%*
	DoF	+8.7%	
	$(A + \text{DoF})/2$	**+9.1%**	
	A	**+9.4%**	
Hand Pose	DoF/2	+287.5%	*+122.7%*
	DoF	**+98.3%**	
	$(A + \text{DoF})/2$	**+77.7%**	
	A	**+27.3%**	
Leg Walk	DoF/2	+9.3%	*+11.8%*
	DoF	**+18.0%**	
	$(A + \text{DoF})/2$	+3.4%	
	A	**+16.6%**	
Rough Terrain	DoF/2	+13.1%	*+31.3%*
	DoF	**+60.2%**	
	$(A + \text{DoF})/2$	**+25.3%**	
	A	**+26.4%**	
Stair Terrain	DoF/2	+24.3%	*+25.5%*
	DoF	**+27.8%**	
	$(A + \text{DoF})/2$	**+25.5%**	
	A	**+24.4%**	

D.3 Decoder conditioning

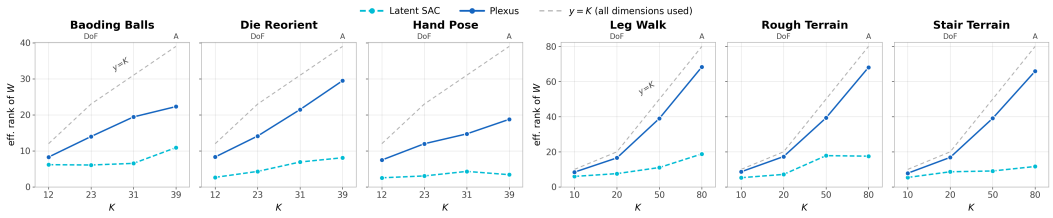

Figure 5: Effective rank of W at convergence against latent width, with $y = K$ marking a basis using all the directions it was given. *Solid:* Plexus. *Dashed:* Latent SAC.

Figure 5 reads the basis against the $y = K$ diagonal, where a decoder uses every direction it was given. Plexus tracks that diagonal at a roughly constant fraction whatever the width, where Latent SAC falls further below it as K grows.

Figure 6 gives three scalar summaries of the same geometry, each against K: effective rank, $\log\det(W^\top W)$ and $\log\sigma_{\min}(W)$. They are the converged form of what Figure 3 follows through training, and easier to compare across configurations. Plexus sits above Latent SAC on all three, on every task and at every width, and the gap widens with the latent: more directions mean more places for the uncorrected arm to collapse onto, where ours stays spread.

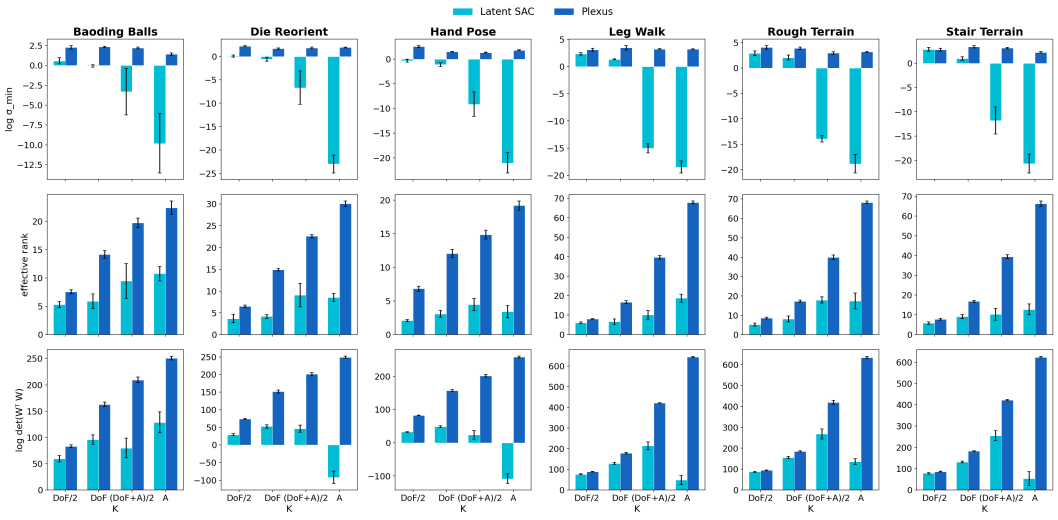

Figure 6: Decoder conditioning summaries: effective rank, $\log\det(W^\top W)$, and $\log\sigma_{\min}(W)$ as a function of K, per task and per body.

D.4 Decoder initialisation and freezing

Table 8: Decoder initialisation and freezing, all at variant (d). The random-init arm sits at its own DoF setting; the DynSyn arms sit at its own synergy count. Best score per task in bold, with * where it exceeds every other arm under a one-sided Welch test at $p < 0.05$.

Task	Decoder	K	Score	rank/K
Baoding Balls	random init	23	**0.641** ±0.080	0.61
	DynSyn init	25	0.551 ±0.090	0.66
	DynSyn init, frozen	25	0.223 ±0.082	0.64
	DynSyn itself	25	0.575 ±0.072	–
Die Reorient	random init	23	0.763 ±0.009	0.61
	DynSyn init	25	**0.804** ±0.040	0.68
	DynSyn init, frozen	25	0.621 ±0.029	0.64
	DynSyn itself	25	0.582 ±0.071	–
Hand Pose	random init	23	0.511 ±0.020	0.52
	DynSyn init	25	**0.577** ±0.043	0.56
	DynSyn init, frozen	25	0.000 ±0.000	0.64
	DynSyn itself	25	0.033 ±0.011	–
Leg Walk	random init	20	17,813 ±732	0.83
	DynSyn init	20	**21,311** ±387*	0.80
	DynSyn init, frozen	20	327 ±5	0.84
	DynSyn itself	20	17,357 ±922	–
Rough Terrain	random init	20	14,378 ±947	0.86
	DynSyn init	20	14,216 ±923	0.81
	DynSyn init, frozen	20	224 ±3	0.84
	DynSyn itself	20	**16,457** ±1,018	–
Stair Terrain	random init	20	11,413 ±1,212	0.84
	DynSyn init	20	**12,779** ±680	0.77
	DynSyn init, frozen	20	275 ±6	0.84
	DynSyn itself	20	12,740 ±1,122	–

D.5 TRAINING CURVES

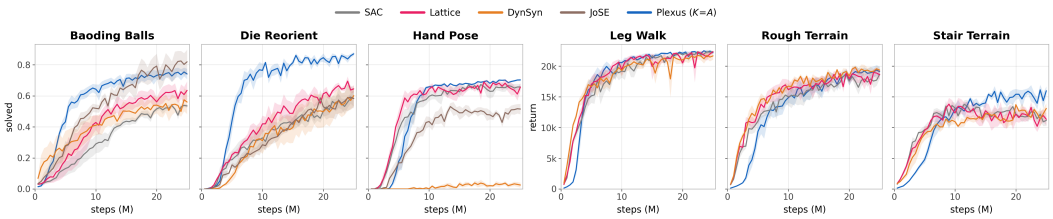

Figure 7: Baseline training curves on each task over 25×10^6 steps, each baseline at the configuration it is scored at in Figure 2. Plexus at $K = A$ is drawn behind them for reference. Curves show the mean across 5 seeds; shaded bands are SEM.

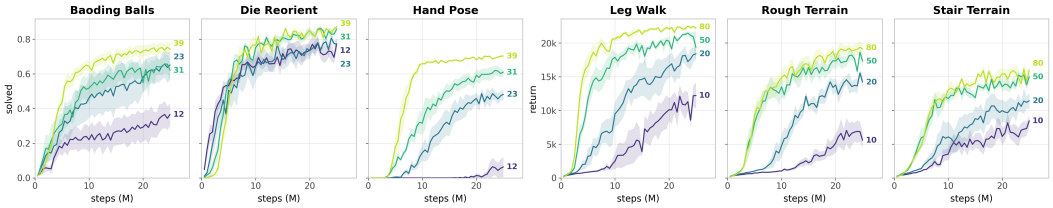

Figure 8: Plexus training curves on each task at all four latent dimensions $K \in \{\text{DoF}/2,\ \text{DoF},\ (\text{DoF} + A)/2,\ A\}$. Same conventions as Figure 7.

Figures 7 and 8 show the training curves for all baselines as well as Plexus at all evaluated latent widths. Most of the curves show a steep increase within the first 10M steps then start plateauing with marginal gains. At $K = A$, Plexus consistently reaches similar or higher performance, and generally higher K values almost always reach a better final score and converge faster, while lower K starts ramping later and plateaus lower within the budget. The curves however suggest that some of the lower-K runs have not fully converged at the time we report, so the gaps at the smallest K should be read as upper bounds on the cost of compression.

D.6 GRADIENT PATH AGAINST TEMPERATURE

Table 9: Gradient path against temperature, on Baoding and Leg Walk tasks, run at every variant (Table 3): (a) latent entropy, (b) stop-gradient on W_{raw}, (c) (b) with α pinned at 0.05 instead of autotuned, (d) full. $\log\det(J^\top J)$ is blank for (a), which never computes the term. A cell reading *diverged* lost all five of its seeds. Rank and $\log\det(J^\top J)$ are kept for those cells, since they are what the collapse looks like. Every other cell scores all five seeds.

		Score		rank/K		$\log\det(J^\top J)$	
Variant	K	Baoding	Leg Walk	Baoding	Leg Walk	Baoding	Leg Walk
(a)	DoF	0.475 $\pm$0.101	15,099 $\pm$1,179	0.26	0.38	–	–
	$\frac{A+\text{DoF}}{2}$	0.439 $\pm$0.063	20,475 $\pm$696	0.21	0.22	–	–
(b)	DoF	*diverged*	19,671 $\pm$844	0.05	0.67	-78	$+54$
	$\frac{A+\text{DoF}}{2}$	*diverged*	*diverged*	0.05	0.03	-84	-56
(c)	DoF	0.284 $\pm$0.094	18,923 $\pm$829	0.30	0.71	$+21$	$+56$
	$\frac{A+\text{DoF}}{2}$	0.552 $\pm$0.034	18,620 $\pm$1,203	0.25	0.30	$+13$	$+83$
(d)	DoF	0.641 $\pm$0.080	17,813 $\pm$732	0.61	0.83	$+67$	$+60$
	$\frac{A+\text{DoF}}{2}$	0.650 $\pm$0.055	21,169 $\pm$457	0.63	0.78	$+82$	$+142$

D.7 CASE STUDIES AT SMALL K

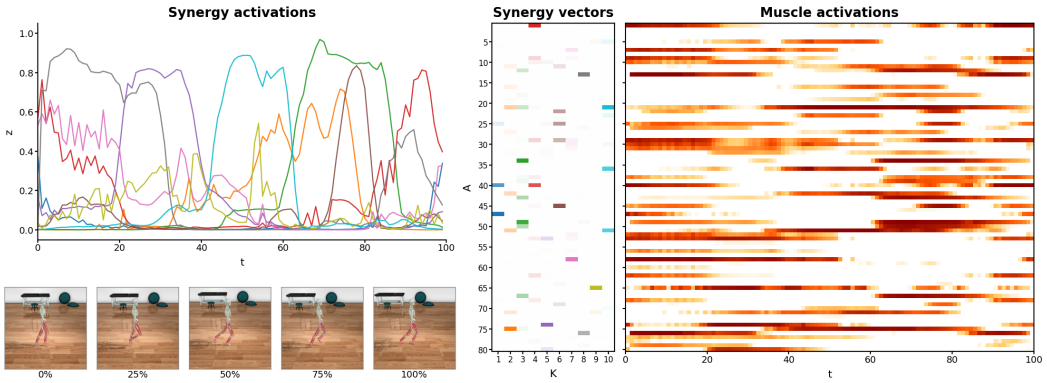

Figure 9: Leg Walk at K=DoF/2=10 with an unconstrained decoder, best-performing seed. *Left:* latent trajectory $z(t)$ over one averaged stride (left heel-strike to left heel-strike), with snapshots at 0%, 25%, 50%, 75% and 100% of the cycle. *Centre:* column-normalised synergy matrix W. *Right:* the corresponding muscle activations $a(t)$ over the same averaged stride, in the row order of W.

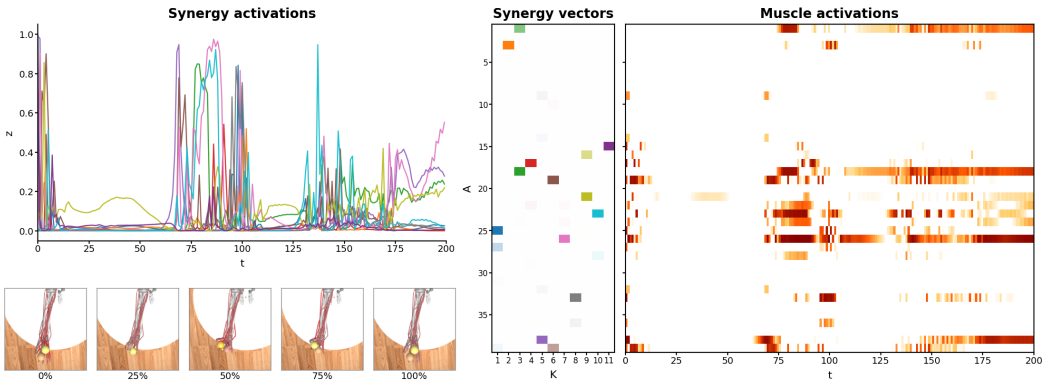

Figure 10: Baoding ball rotation at K=11 on MyoHand (A=39, DoF=23). Same conventions as Figure 9.

Figure 9 is the unconstrained counterpart to the tied decoder of Figure 4. It shows how the policy has decomposed the 80-dimensional muscle space into 10 sparse, anatomically structured synergies where each column of W recruits distinct muscles. The latent trajectory $z(t)$ exhibits clear temporal structure: different synergies are recruited sequentially over the gait cycle, with most dimensions showing a single broad activation peak. In particular, a shift occurs around $t \approx 40$–50%, as several synergies simultaneously ramp up or down, corresponding to the transition from stance (foot in contact with the ground) to swing (foot off the ground). A symmetric transition is observable near $t \approx 90$–100%, corresponding to the analogous event on the opposite leg. The synergy matrix W itself is sparse and anatomically structured: for instance, synergy k=5 groups left *soleus* and *lateral gastrocnemius*: two ankle plantarflexors jointly recruited during terminal stance.

Manipulation in Figure 10 looks different. Unlike locomotion, Baoding ball rotation does not impose a strict movement cycle, and the latent trajectory $z(t)$ accordingly lacks the sequential, wave-like synergy recruitment seen on the leg. Instead, the policy issues brief, sparse bursts of activity in highly specific muscle subsets: individual synergies are transiently recruited in isolation rather than cycling through them in a phase-locked pattern. This reflects the fine-grained, contact-driven nature of in-hand manipulation: precise force regulation on the two balls requires targeted co-activation of small, task-specific muscle groups (visible as concentrated columns in W) rather than the broad rhythmic contractions characteristic of locomotion.

E ABLATIONS

E.1 GSDE ABLATION

The main results use independent Gaussian exploration noise resampled at every environment step (re-sampling frequency 1, "gSDE off"). gSDE (Raffin et al., 2022) replaces this with state-dependent noise resampled less frequently, here every 4 steps ("gSDE on"). We re-run Plexus at $K = A$, full-dim SAC, and Lattice with gSDE enabled on all six tasks, keeping every other hyperparameter fixed to the values in Table 4.

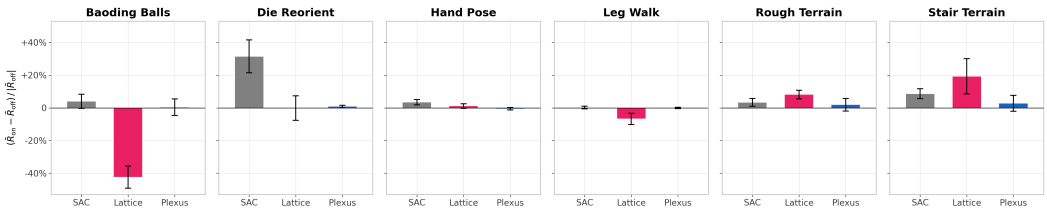

Figure 11: Relative impact of gSDE on final score: $(\bar{R}_{\text{on}} - \bar{R}_{\text{off}})/|\bar{R}_{\text{off}}|$ per task and method (5 seeds). Positive bars indicate gSDE helps, negative bars indicate gSDE hurts. For Plexus we report $K = A$.

Fig. 11 shows that Plexus's performance is essentially unchanged by gSDE: differences are within seed noise on every task. gSDE makes the perturbation a smooth function of the state and holds it fixed over several steps. Acting through W already confines every perturbation to the K-dimensional synergy manifold, so exploration is coordinated across muscles by construction rather than independent per actuator; what gSDE adds on top is temporal persistence alone, and that changes little here. Lattice sees the largest gSDE effect but it is mixed: it benefits on the hardest leg tasks (Rough and Stair Terrain) but degrades on Baoding and Leg Walk. Full-dim SAC sees a clear gain on Die Reorient and negligible or modest improvements on the harder leg tasks.

E.2 TARGET ENTROPY $\bar{\mathcal{H}}$ AND CHOLESKY ε ABLATIONS

We jointly report two Plexus-only sensitivity studies on Leg Walk and Baoding at $K = \text{DoF}$ ($K = 20$ and $K = 23$ respectively), with 5 seeds per setting and all other hyperparameters fixed to the values in Table 4. The default target entropy $\bar{\mathcal{H}} = -K$ mirrors SAC's standard $\bar{\mathcal{H}} = -A$ rule applied to the K-dimensional latent action; we sweep $\bar{\mathcal{H}} \in \{-K/2, -K, -A\}$. The Cholesky regularizer ε guards against ill-conditioning early in training; we sweep $\varepsilon \in \{10^{-3}, 10^{-4}, 10^{-8}\}$.

Table 10: Plexus at $K = \text{DoF}$ on Leg Walk and Baoding under varying target entropy $\bar{\mathcal{H}}$ (left) and Cholesky regularizer ε (right). Each cell reports the metric (return for Leg Walk, fraction solved for Baoding) as mean $\pm$ s.e.m.over 5 seeds. The middle row of each sweep is the default used in the main paper.

(a) Target entropy $\bar{\mathcal{H}}$ sweep.

$\bar{\mathcal{H}}$	**Leg Walk** (return)	**Baoding** (solved)
$-K/2$	$18,178 \pm 1,382$	0.629 ± 0.043
$-K$	$19,083 \pm 952$	0.663 ± 0.076
$-A$	$18,014 \pm 2,489$	0.725 ± 0.075

(b) Cholesky regularizer ε sweep.

ε	**Leg Walk** (return)	**Baoding** (solved)
10^{-3}	$20,179 \pm 734$	0.739 ± 0.055
10^{-4}	$19,083 \pm 952$	0.663 ± 0.076
10^{-8}	$18,189 \pm 1,595$	0.618 ± 0.090

Table 10 shows that Plexus is robust to both knobs across two qualitatively different tasks. For both target entropy and the Cholesky regularizer, all three settings overlap within seed noise on each task. These are a separate set of runs from the main sweep, so the default cells are not expected to reproduce the corresponding entries of Tables 5 and 6 exactly. At $K = \text{DoF}$ the Plexus training landscape is intrinsically noisier than at $K = A$ so the wide SEMs in the table reflect environmental seed variance rather than sensitivity to either knob. We therefore retain $\bar{\mathcal{H}} = -K$ and $\varepsilon = 10^{-4}$ as simple, robust defaults.

F WALL-CLOCK

We benchmark training throughput on Leg Walk with all methods on a single consumer-grade GPU (RTX 3070, 8 GB) using num_envs $= 8$ to fit local memory. We measure $\sim$10 000 environment steps after a 2 000-step replay-buffer warm-up. Reported numbers are the steps-per-second logged by the orchestrator at end of run. The results we report in the main paper were obtained on cluster GPUs with num_envs $= 30$. This local benchmark fixes the hardware so that all methods are compared on equal footing.

Table 11: Training throughput on Leg Walk. "Relative to SAC" is the ratio of measured speeds.

Method	CUDA-graph	Speed (env steps/sec)	Relative to SAC
SAC	✓	1226.7	1.00×
DynSyn	no	404.2	0.33×
Lattice	no	309.8	0.25×
Plexus K=10	✓	1169.5	0.95×
Plexus K=20	✓	1173.9	0.96×
Plexus K=50	✓	1136.8	0.93×
Plexus K=80	✓	995.2	0.81×

Plexus and our SAC implementation are end-to-end CUDA-graph captured: the actor and critic update steps are wrapped in `tensordict.nn.CudaGraphModule`, giving a 3–4× speed-up over eager mode. Lattice and DynSyn cannot be captured natively: DynSyn's per-step ramp schedule on the synergy weight matrix involves data-dependent control flow and as stated in the original Lattice paper (Chiappa et al., 2023), multivariate Gaussian with full covariance matrix adds a significant additional computational cost. Results are available in Table 11.

On the other hand, Plexus's throughput is comparable to vanilla SAC across all K, confirming that the Cholesky-based Jacobian correction adds negligible overhead on top of the latent decoder when both are CUDA-graph captured. The only visible cost is at $K = A$, where throughput falls to $0.81\times$, reflecting both the larger decoder matrix-vector product and the larger $K \times K$ Cholesky in the actor update.

Cluster runs. The main results in Sec. 5 were produced on a shared Slurm-managed cluster, one job per (method, task, K, seed) with num_envs $= 30$, requesting one GPU with at least 20 GB of VRAM, 30 CPU cores, and 32 GB of RAM. The cluster pool is heterogeneous: NVIDIA RTX 3090 24 GB, RTX A5000 24 GB, A100 PCIe 40 GB, A100 PCIe 80 GB and RTX A6000 48 GB. The same job script is dispatched regardless of which GPU is allocated.

A full 25×10^6-step run takes $\sim 4 - 5$ h for Plexus, Latent SAC, and full-dim SAC depending on the task, and $\sim 11 - 12$ h for the others, consistent with the controlled-throughput ratios in Table 11. Aggregating over the reported sweep (six tasks, Plexus and Latent SAC at four values of K, SAC and Lattice at one configuration each, DynSyn at one configuration on the hand and two on the leg, JoSE on the hand tasks only, five seeds per configuration) the main results total roughly $3{,}000$ GPU-hours. Preliminary experiments (architecture exploration before the final decoder design was fixed, sanity runs on smaller environments) as well as necessary ablations account for an additional $\sim$1,500 GPU-hours.