EXPAND: EXTENSIBLE POLICY CAPACITY FOR UNSUPERVISED SKILL DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised reinforcement learning pretrains agents without task rewards for downstream learning and adaptation. Active exploration broadens state coverage but often lacks a reusable skill interface; skill discovery learns controllable behaviors whose coverage can plateau. Fixed policy capacity may limit exploration even as intrinsic objectives and representations improve. We introduce **EX**tensible **P**olicy c**A**pacity for u**N**supervised skill **D**iscovery (EXPAND), a framework that adds skill-conditioned cascade stages during pretraining. Stages train jointly under the original objective; gates conditioned on state and skill recursively combine action distribution parameters. EXPAND increases METRA's final historical coverage by **71.1%** on average across seven environments and generally outperforms parameter-matched wider, deeper, and parallel gated policies. EXPAND also improves coverage with LSD and DIAYN, with gains varying by environment and metric. EXPAND also improves METRA's overall zero-shot goal reaching and hierarchical control with frozen skills.

1 INTRODUCTION

Unsupervised reinforcement learning (URL) pretrains transferable behaviors, representations, or environment models without task rewards to reduce downstream interaction. Its three main paradigms are active exploration for state coverage, skill discovery for behaviors controlled by latent variables, and dynamics pretraining that learns reusable world models or representations for planning and control (Liu & Abbeel, 2021; Eysenbach et al., 2019; Sekar et al., 2020). Contrastive RL uses a contrastive objective to learn state-action and future-state representations of goal reachability (Eysenbach et al., 2022). We focus on the first two paradigms to combine broad exploration with reusable behavior.

Active exploration rewards visits to underexplored regions using state-visitation diversity or proxies such as novelty and information gain (Liu & Abbeel, 2021; Bellemare et al., 2016; Pathak et al., 2017; 2019; Houthooft et al., 2016). This collects diverse experience before downstream tasks are known. Exploration policies often lack an explicit skill-selection interface, so goal-conditioned or hierarchical control still requires additional controller learning or policy adaptation, even after broad state coverage.

Skill discovery provides a reuse interface through a policy $\pi_\theta(a \mid s, z)$ conditioned on latent skills z. Mutual-information objectives encourage dependence between skills and states or transitions, producing distinguishable behaviors (Eysenbach et al., 2019; Sharma et al., 2020). Distinct skills can still remain within visited regions and add little coverage. METRA improves exploration through a latent metric reflecting environment transitions (Park et al., 2024), but exploration can remain local, limiting the behavioral choices available for downstream control over larger areas.

Both paradigms rely on policies to turn intrinsic signals into behavior, usually with a fixed architecture throughout pretraining. Policy-induced visitation shapes the data used for subsequent updates. Changing intrinsic objectives can select different behaviors within this policy family but cannot enlarge its representable trajectory distributions. We therefore hypothesize that fixed policy capacity can bottleneck exploration, and test cascade organization under fixed skill-discovery objectives and interaction budgets.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

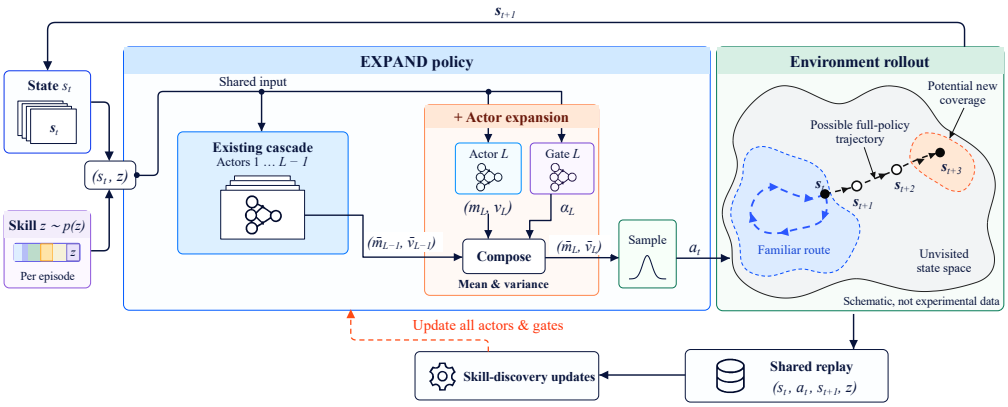


Figure 1: **EXPAND framework for skill-conditioned policy cascades.** Each cascade stage receives the current state s_t and a skill variable z sampled once per episode. The gate recursively combines the action distribution parameters of the new cascade stage and the existing cascade to form the policy used to sample a_t. We store environment transitions in a shared replay buffer and use them to update all cascade stages and gates. The trajectories on the right illustrate potential coverage gains after policy expansion and do not depict experimental results.

We introduce **EX**tensible **P**olicy c**A**pacity for u**N**supervised skill **D**iscovery (EXPAND), a plug-in policy capacity expansion framework for unsupervised skill discovery. Figure 1 shows how EXPAND builds on Cascade (Schmöcker & Dockhorn, 2025): new stages add capacity, and gates conditioned on state and skill recursively combine stage outputs into a single action distribution. EXPAND changes policy parameterization while preserving the base method's intrinsic objective and skill interface. Paired comparisons with and without EXPAND measure its incremental effect.

We evaluate position coverage in locomotion and navigation environments, using METRA as the main instantiation. Rerunning cascade prefixes at the same final checkpoint reveals regions first visited by longer prefixes. EXPAND improves all three coverage AUCs for LSD and DIAYN on Cheetah and Quadruped; Walker results vary by method and metric. Within METRA, parameter-matched comparisons generally favor cascades over wider, deeper, and parallel gated policies. With EXPAND, METRA's pretrained skills support zero-shot goal-conditioned control and hierarchical control with frozen skills. Ablations show that stages and gate inputs affect coverage. Within the tested range, greater final cascade depth generally broadens coverage, with plateau depths varying by environment; expansion timing has little effect at fixed final depth. These results point to policy capacity organization and final depth as exploration design choices.

Our contributions are:

1. We bound occupancy perturbations and, under regularity conditions, relate uncovered occupancy to remaining parameter drift. We derive conditions for local coverage gains from expansion and skill-conditioned gating.
2. We introduce EXPAND, a framework for unsupervised skill discovery that expands policy cascades while preserving the base method's intrinsic objective and skill interface.
3. Paired evaluations assess coverage gains on METRA, LSD, and DIAYN; parameter-matched controls and downstream tests on METRA assess policy organization and skill reuse. Ablations examine actor structure, gating, and expansion timing.

2 Related Work

Unsupervised exploration and skill discovery. APT constructs intrinsic rewards from learned representations (Liu & Abbeel, 2021). DIAYN uses mutual information, while LSD and METRA constrain learned representations to discover skills (Eysenbach et al., 2019; Park et al., 2022; 2024). EXPAND expands skill-conditioned policies while preserving the intrinsic objective and skill interface. Paired evaluations on METRA, LSD, and DIAYN measure this policy replacement's incremental effect.

Policy capacity and multi-policy structures. Neuroplastic Expansion and Forget-and-Grow expand networks to mitigate plasticity loss (Liu et al., 2025; Kang et al., 2025), while SimBa improves training of large actor-critic networks (Lee et al., 2025). Cascade uses sequential ensembling for continuous control (Schmöcker & Dockhorn, 2025); EXPAND adapts this architecture to skill-conditioned policies. CeSD provides an algorithm-level reference with changes to skill learning and value ensembles (Bai et al., 2024), while our MoP control mixes action densities in parallel. Appendix A gives the detailed discussion.

3 Preliminaries and Problem Formulation

We consider a finite-horizon MDP without extrinsic rewards, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, P, H)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, initial distribution $\mu \in \Delta(\mathcal{S})$, transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and episode length H. A conditional policy $\pi_\theta(a \mid s, z)$ uses a latent variable $z \sim p(z)$ sampled once per episode. Skill discovery uses discrete or continuous $z \in \mathcal{Z}$; active exploration without a skill interface corresponds to $\mathcal{Z} = \{z_0\}$. The joint distribution of skills and trajectories is

$$q_\theta(\tau, z) = p(z)\mu(s_0) \prod_{h=0}^{H-1} \pi_\theta(a_h \mid s_h, z) P(s_{h+1} \mid s_h, a_h), \tag{1}$$

where $\tau = (s_0, a_0, \ldots, s_{H-1}, a_{H-1}, s_H)$. A cascade stage is a policy component in Cascade; i indexes stages and m denotes their current number. We use k to index the collection of a fresh trajectory, c for a collection-and-update cycle of the base method, and t for cumulative environment interactions. In cycle c, a novelty model, skill discriminator, or representation ξ_c supplies the intrinsic signal. Holding ξ_c fixed gives the policy objective

$$\mathcal{J}_c(\theta) = \mathbb{E}_{(\tau,z)\sim q_\theta} \left[\sum_{h=0}^{H-1} r^{\text{int}}_{\xi_c}(s_h, a_h, s_{h+1}, z) \right]. \tag{2}$$

Active exploration derives this signal from visitation frequency, prediction error, or uncertainty; skill discovery encourages dependence between skills and states or transitions. Both rely on the policy to translate these signals into behavior.

4 Policy Changes and State Occupancy

Continued trajectory collection need not yield continued exploration of new regions. Figure 2 illustrates this behavior: the policy collects its own training data, and subsequent trajectories repeatedly traverse visited regions, yielding progressively smaller coverage increments. A perturbation bound relates policy changes to state visitation.

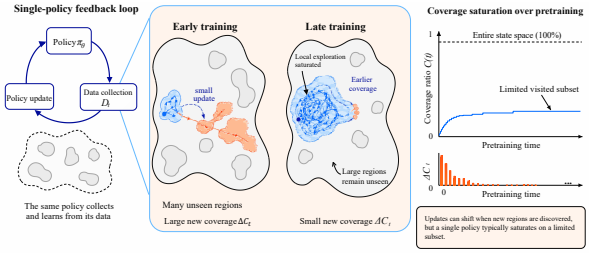


Figure 2: **Schematic of cumulative coverage.** The policy collects its own training data (left) and revisits covered regions (center). Blue, orange, and gray denote previously covered, newly covered, and unvisited regions. Coverage is bounded on a fixed finite partition (right), though unvisited regions may remain reachable within the policy family.

For any policy π, let q_π be the joint skill–trajectory distribution obtained by replacing π_θ with π in Equation 1; hence $q_{\pi_\theta} = q_\theta$. For every measurable state set $B \subseteq \mathcal{S}$, define time-averaged state occupancy as

$$\bar{d}_{q_\pi}(B) = \frac{1}{H+1} \sum_{h=0}^{H} \Pr_{q_\pi}(s_h \in B), \tag{3}$$

For probability measures ν and ν' on a common measurable space, $d_{\text{TV}}(\nu, \nu') = \sup_A |\nu(A) - \nu'(A)|$ denotes their total variation distance, where the supremum ranges over measurable sets A.

Proposition 1 (**Occupancy perturbation bound**). Under common μ, P, $p(z)$, and H, suppose that two policies $\pi, \widetilde{\pi}$ satisfy

$$\sup_{s,z} d_{\mathrm{TV}}(\pi(\cdot \mid s, z), \widetilde{\pi}(\cdot \mid s, z)) \leq \varepsilon, \qquad \varepsilon \in [0, 1].$$

Then

$$d_{\mathrm{TV}}(q_\pi, q_{\widetilde{\pi}}) \leq H\varepsilon, \qquad d_{\mathrm{TV}}(\bar{d}_{q_\pi}, \bar{d}_{q_{\widetilde{\pi}}}) \leq \frac{H}{2}\varepsilon. \tag{4}$$

The factor $H/2$ follows from averaging the state-marginal differences over steps 0 through H; Appendix B.2 gives the proof. The bound applies across policy parameterizations and characterizes the magnitude of occupancy changes; coverage increments evaluate exploration gains separately.

Appendix B.3 shows that, under cumulative counting on a fixed finite partition, expected coverage increments vanish for any policy sequence satisfying the sampling convention, including parameter sequences before convergence and sequences with changing architectures. This counting limit characterizes asymptotic coverage increments; actual training sequences and fixed-budget controls identify capacity bottlenecks and finite-budget plateau timing.

5 Cascade Expansion of Skill-Conditioned Policies

EXPAND adds trainable stages during pretraining and updates them jointly with the existing stages. Figure 3(a) and (b) illustrate the process: changes in the full policy's action sequence may allow a trajectory to leave repeatedly visited regions, producing the coverage growth depicted in Figure 3(c). The action distribution depends on how the new stage is combined with the existing cascade.

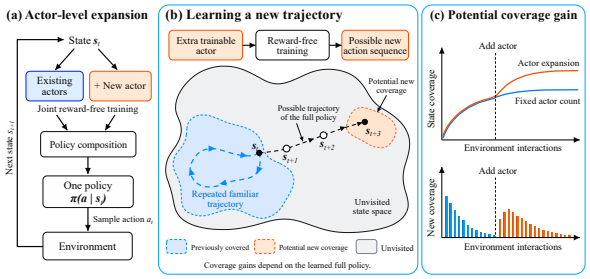


Figure 3: **Schematic of actor expansion.** (a) An actor is added and all stages train jointly. (b) New action sequences can reach orange regions from visited blue regions; gray denotes other unvisited regions. (c) Actor additions (dashed lines) may temporarily increase coverage increments. Upper curves show fixed (blue) and increasing (orange) actor counts.

5.1 Cascade Parameterization and Distribution Changes

EXPAND uses the Cascade architecture (Schmöcker & Dockhorn, 2025).

All stages and gates receive the same input (s, z), while the skill space $\mathcal{Z}$, prior $p(z)$, and intrinsic objective remain unchanged. A skill $z \sim p(z)$ is held fixed throughout each episode. Stage i outputs a mean $\mu_i(s, z)$ and standard deviation $\sigma_i(s, z)$. For $i \geq 2$, the same network also outputs a learnable weight $\lambda_i(s, z)$ for the existing prefix. The new stage receives the complementary weight $1 - \lambda_i$.

Suppressing (s, z), let $v_i = \sigma_i^2$ and $(\mu^{[1]}, v^{[1]}) = (\mu_1, v_1)$. Here $(\mu^{[i]}, v^{[i]})$ denotes the aggregate mean and coordinatewise variance after composing the first i stages; the bracketed superscript is a stage index rather than an exponent. The Gaussian parameters are combined recursively:

$$\mu^{[i]} = \lambda_i \mu^{[i-1]} + (1 - \lambda_i)\mu_i, \tag{5}$$

$$v^{[i]} = \lambda_i^2 v^{[i-1]} + (1 - \lambda_i)^2 v_i. \tag{6}$$

Squares and products are coordinatewise, and the scalar gate is shared across action dimensions. A policy with m stages draws one sample from the final distribution:

$$u \sim \mathcal{N}\left(\mu^{[m]}, \mathrm{diag}(v^{[m]})\right), \qquad a = \tanh(u). \tag{7}$$

Here u is the Gaussian action variable before the $\tanh$ transformation, and $\mathrm{diag}(v^{[m]})$ is the covariance matrix whose diagonal is $v^{[m]}$. Training uses this single tanh-transformed Gaussian and its log density (Haarnoja et al., 2018a).

The gate head uses Xavier-normal weights (Glorot & Bengio, 2010) and zero bias, and the variance-compensation term is zero at expansion. If $\lambda_i = 1/2$ and the new stage has the same variance

as the full existing cascade, the combined variance is halved. Appendix C.1 gives the additional representability and variance-compensation conditions required to preserve the previous distribution exactly; we treat distribution preservation as a property determined by these conditions.

5.2 Conditions for expansion and skill-conditioned gating to improve coverage

The new trajectory in Figure 3(b) must cross the boundary of the visited region. An outward mean shift can increase the crossing probability, whereas variance contraction may reduce it. The following local model quantifies this tradeoff.

Proposition 2 (**Local boundary-crossing condition**). Fix an input (s, z) and the existing cascade. In a one-step model in the Euclidean latent space before the $\tanh$ transformation, entering an unvisited region is equivalent to $n^\top u \geq b$, where u is the action variable in Equation 7, $n \neq 0$ is the normal vector of a local boundary, and b is its threshold. Let $(\bar{\mu}, \bar{v})$ and $(\mu_{\text{new}}, v_{\text{new}})$ denote the parameters of the existing cascade and new stage, respectively, with λ the weight on the existing prefix. Let $d = b - n^\top \bar{\mu}$, $\delta = n^\top(\mu_{\text{new}} - \bar{\mu})$, $w = n^\top \operatorname{diag}(\bar{v}) n > 0$, and $R = \lambda^2 + (1-\lambda)^2 n^\top \operatorname{diag}(v_{\text{new}}) n / w$. These quantities denote, respectively, the existing mean's projected deficit to the boundary, the new stage's projected mean displacement, the existing distribution's projected variance, and the multiplicative change in projected variance after expansion. The crossing probabilities before and after expansion, p_0 and p_+, satisfy

$$p_+ > p_0 \quad \Longleftrightarrow \quad (1-\lambda)\delta > (1-\sqrt{R})d. \tag{8}$$

When $d > 0$ and $R < 1$, the mean shift must exceed the threshold on the right. Copying the prefix mean and variance alone decreases the crossing probability. Separate interventions on the mean and variance at a fixed checkpoint can test this prediction. The proposition characterizes local one-step coverage; for general transitions, boundary-approximation error and long-horizon dynamics jointly determine long-horizon coverage growth. Appendix C.1 gives the derivation and limitations.

Skill inputs can help when different skills favor different weights at the same state. Fix the stage outputs, target region, and a joint state–skill evaluation distribution ν. Choose two distinct feasible gate values λ_A and λ_B for the existing prefix, and denote the corresponding one-step reaching probabilities by $p_A(s, z)$ and $p_B(s, z)$. In what follows, $\mathbb{E}_s$ is with respect to the state marginal of ν, while $\mathbb{E}(\cdot \mid s)$ is with respect to its conditional skill distribution given s.

Proposition 3 (**Value of skill-conditioned gating**). Let $D(s, z) = p_A(s, z) - p_B(s, z)$, the one-step reaching-probability advantage of gate A over gate B. Let $V^*_{s,z}$ and V^*_s denote the optimal average reaching probabilities when selection between these two gate values may depend on (s, z) or on s alone, respectively. Then

$$V^*_{s,z} - V^*_s = \frac{1}{2}\mathbb{E}_s[\mathbb{E}(|D| \mid s) - |\mathbb{E}(D \mid s)|] \geq 0. \tag{9}$$

The gain is strict if and only if, on a set of states of positive measure under the state marginal of ν, both $\Pr_\nu(D > 0 \mid s)$ and $\Pr_\nu(D < 0 \mid s)$ are positive, where $\Pr_\nu$ denotes conditional probability under ν.

This result applies to general measurable boundaries while fixing the evaluation distribution and stage outputs. The additional value of optimal gate selection comes from opposing preferences across skills. A frozen-stage gate-preference experiment can measure how much of this value is realized by neural gating and joint training. Appendix C.2 gives the proof.

5.3 Expansion during pretraining and skill reuse

Let USD denote the base skill-discovery method, ψ its auxiliary parameters and training state, and $\mathcal{D}$ its replay buffer or trajectory batch. Let θ collect the parameters of all stages and gates, and let $\pi_\theta^{[m]}$ denote the full policy obtained by composing the first m stages according to Equations 5–7. EXPAND preserves the base method's collection and update order, jointly training all stages and gates at each actor update. A new stage is added at cycles $c \in \mathcal{E}_{\text{grow}}$, up to M stages, and its parameters are registered with the actor optimizer. Algorithm 1 in Appendix C.3 summarizes pretraining; the appendix specifies initialization and how cycles map to interaction steps.

After pretraining, the full policy is frozen. Downstream control uses the base method's goal-to-skill mapping or a high-level policy that selects z. Joint pretraining may change the behavior associated with a given z, so downstream skill reuse and preservation of prior behavior at expansion require separate evaluation.

6 EXPERIMENTS

We organize the evaluation of EXPAND around four research questions: (1) **RQ1 (Exploration and applicability):** Can EXPAND improve Policy, Replay, and History coverage across environments and skill discovery methods under the same environment interaction budget? (2) **RQ2 (Structural contribution):** Can cascade organization achieve higher coverage than other policy structures at matched parameter counts, and how do individual stages and skill-conditioned gates contribute? (3) **RQ3 (Skill reuse):** Can EXPAND's pretrained skills improve zero-shot goal reaching and hierarchical control with frozen skills? (4) **RQ4 (Configuration sensitivity):** How sensitive is position coverage to actor depth within each stage and expansion timing?

6.1 EXPERIMENTAL SETUP

Environments and evaluation. We evaluate OGBench AntMaze Giant and Large (Park et al., 2025), and DMC Cheetah, Walker, Quadruped, Humanoid, and Fish (Tassa et al., 2018). Comparisons use matched interaction budgets and evaluation protocols, with intrinsic rewards only. All experiments use eight random seeds except the stage-contribution ablation. We report zero-shot goal reaching on Cheetah, Walker, and Quadruped, and hierarchical control on three tasks each in Cheetah and Walker. Appendix D.1 gives the protocol.

Comparison methods. We compare METRA with METRA+EXPAND, and compare LSD and DIAYN with their respective EXPAND variants on Cheetah, Quadruped, and Walker. CeSD, APT, and AptMetra represent alternative exploration designs. Wider networks, deeper SimBa networks, and a Mixture of Policies (MoP) compare parameter scale and policy organization. Flat Gaussian Aggregation tests recursive aggregation against a softmax weighting of the same Gaussian stage outputs. Appendix E.2 describes its implementation.

Downstream evaluation. We report goal-reaching success and final goal distance. During APT pretraining, we train an additional goal-conditioned policy on its replay buffer using Contrastive RL (Eysenbach et al., 2022) and freeze it for evaluation. For hierarchical control, we freeze the low-level skill policy and train a high-level PPO policy (Schulman et al., 2017) to select z under the same downstream interaction budget. Appendices D.5 and D.6 give the protocols.

6.2 MAIN RESULTS

Coverage during pretraining. EXPAND METRA exceeds METRA in final History, Replay, and Policy coverage across all seven environments. Figure 4 shows that it achieves the highest final History coverage in five, with continued late growth on Quadruped and Humanoid; AptMetra leads on AntMaze Large, and APT narrowly leads on Fish. Appendix D.3 shows that it achieves the highest final Replay coverage in all seven and Policy coverage in six, excluding AntMaze Large.

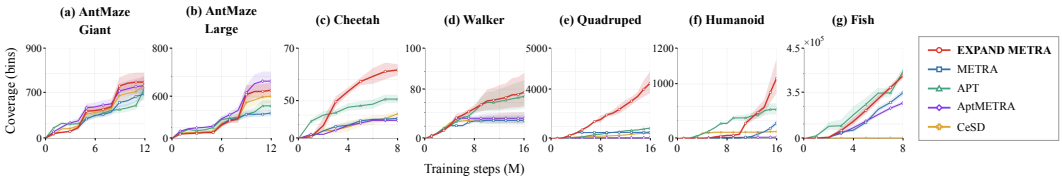


Figure 4: **Pretraining History coverage across seven environments.** We count distinct position cells visited up to each training step. The horizontal axis shows environment interaction steps. Vertical axes use nonlinear scales, with ticks in actual cell counts.

Parameter-matched controls. Figure 5 shows that, with matched parameter counts, EXPAND METRA achieves the highest final History coverage in all five environments, with larger gaps on Quadruped and Humanoid. This supports exploration gains from cascade organization at comparable capacity. Appendix D.4 reports Policy and Replay 300k coverage.

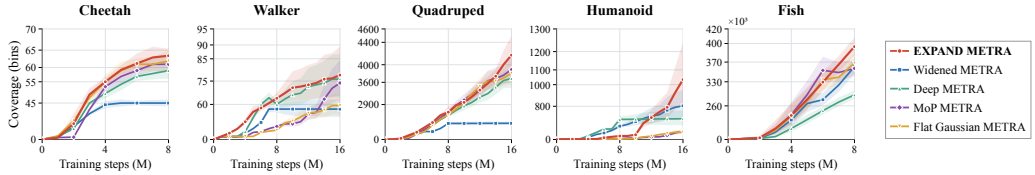


Figure 5: **History coverage with matched parameter counts:** cumulative visited cells over training.

Zero-shot goal reaching. EXPAND METRA attains near-perfect success early on Cheetah and maintains high success at most later checkpoints; Walker success rises with pretraining. Figure 6 shows its lead over the baselines at late checkpoints in both environments. On Quadruped, it has lower goal distance than METRA at most checkpoints but no consistent success advantage. At 16M steps, its final goal distance is lower than METRA's in all three environments.

Hierarchical control with frozen skills. Figure 7 shows that, with frozen skills and a shared budget of 500k downstream interaction steps, EXPAND METRA achieves the highest final mean return on all six Cheetah and Walker tasks. These results support reuse of its pretrained skills for hierarchical control. We exclude APT because it lacks a skill interface.

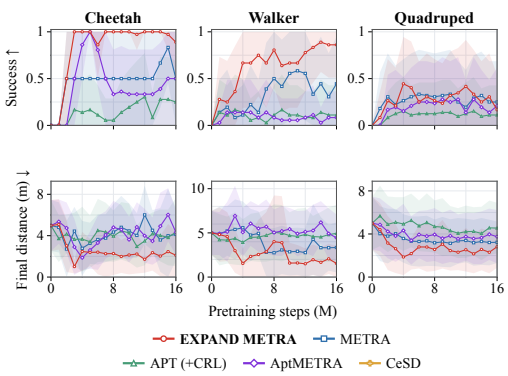


Figure 6: **Zero-shot goal reaching.** Top: success; bottom: final distance in metres.See Appendix D.5 for APT (+CRL).

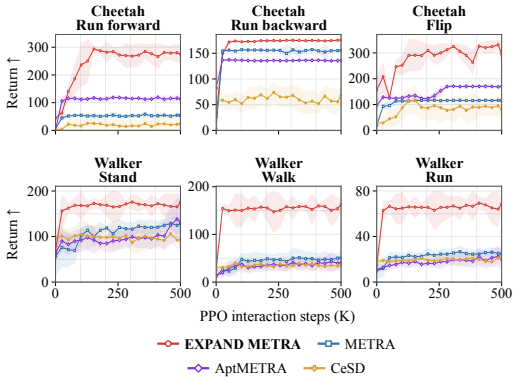


Figure 7: **Hierarchical control with frozen skills.** Mean return versus downstream PPO steps.

Portability beyond the METRA instantiation. Table 1 shows that adding EXPAND to LSD or DIAYN improves all three coverage-curve AUCs on Cheetah and Quadruped. On Walker, LSD's Policy and History AUCs decrease, whereas DIAYN's Policy and Replay 300k AUCs improve and its History AUC is unchanged. Together, these paired results show that EXPAND can improve coverage with other skill-discovery methods under the evaluated protocol.

6.3 Ablation studies and sensitivity analysis

Contributions of cascade stages. Figure 8 compares P_1–P_4 from the same final checkpoint. As the prefix grows, some Cheetah and Walker skills extend their coverage intervals at both ends. Quadruped develops new radial branches, Humanoid expands along the outer boundary, and Fish adds several 3D visitation directions. Longer prefixes thus reach additional positions.

Disabling an individual stage in the full cascade reduces coverage, and the change in visited cells locates the affected regions. The prefix visualization identifies positions first reached after adding a stage, while the intervention measures the full policy's coverage dependence on that stage. Together,

Table 1: Area under the position-coverage curves (AUC, %) for LSD and DIAYN with and without EXPAND.

Environment	Curve	LSD		DIAYN	
		Original	+EXPAND	Original	+EXPAND
Cheetah	Policy	55.5%	**72.6%**	14.9%	**18.0%**
	Replay 300k	35.5%	**56.9%**	11.2%	**27.6%**
	History	76.5%	**96.9%**	30.6%	**69.1%**
Quadruped	Policy	35.7%	**46.7%**	23.7%	**31.2%**
	Replay 300k	20.2%	**37.8%**	10.6%	**24.4%**
	History	40.4%	**96.1%**	27.1%	**33.6%**
Walker	Policy	**35.6%**	29.8%	12.4%	**12.8%**
	Replay 300k	45.2%	**45.6%**	11.5%	**13.7%**
	History	**96.9%**	87.2%	20.3%	20.3%

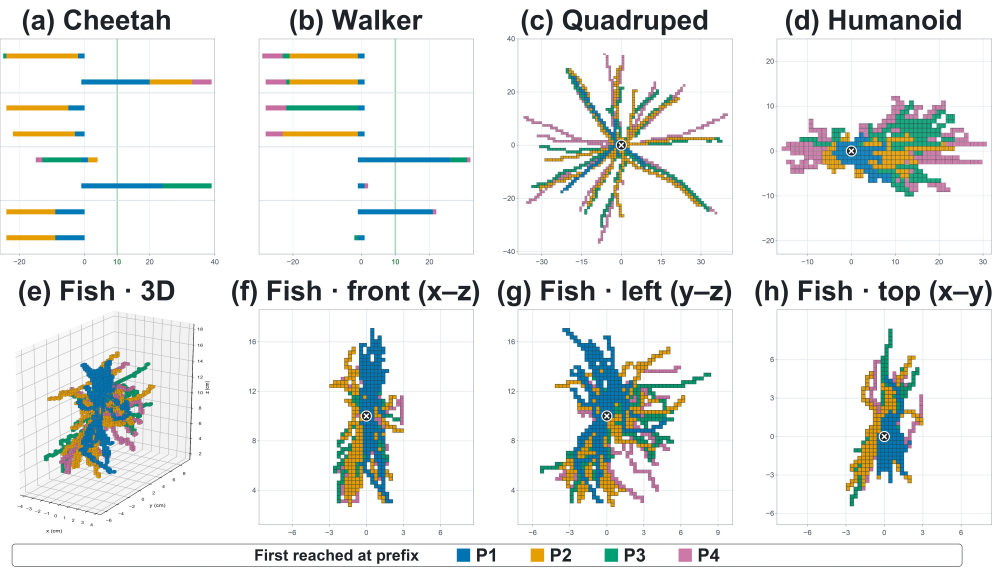


Figure 8: **Prefix coverage at the same final checkpoint.** Pixel colors mark the first of P_1–P_4 to visit each cell, as mapped by the legend. (a,b) Cheetah & Walker; (c,d) Quadruped & Humanoid; (e–h) Fish 3D coverage and its projections.

these analyses answer the RQ2 question of how individual cascade stages affect coverage. This result is consistent with Proposition 2: the theory gives an action-distribution condition under which a new stage raises the local boundary-crossing probability, and the experiment shows the corresponding new regions at the trajectory level. Appendix D.7 describes the prefix construction, trajectory recollection, spatial discretization, and intervention statistics.

Actor structure within each stage. We compare 1, 2, 4, and 8 blocks per actor with four cascade stages and matched actor parameter counts. Figure 9 shows small coverage differences on Cheetah. On Walker, 1 and 8 blocks give higher coverage than 2 and 4 on all three metrics. On Quadruped, one block gives the highest Replay and History coverage, while 4 and 8 blocks give higher, similar Policy coverage. Increasing actor depth therefore gives no monotonic gain at fixed parameter and stage counts; the better configuration depends on the environment and metric.

Gate inputs. Figure 10 shows that (s, z) gating reaches near-maximal coverage on Cheetah by 5M steps and rises on Quadruped. At 16M steps, it exceeds the separately trained s-only and uniform gates in both environments, partly supporting Proposition 3.

Expansion timing. Figure 11 compares fixed additions every 4M steps with three alternatives: four stages from initialization, random addition times, and additions after coverage saturation. Relative final coverage changes range from -4.81% to $+2.98\%$, with no schedule leading across all environments and metrics. Coverage has low timing sensitivity at this budget and final depth.

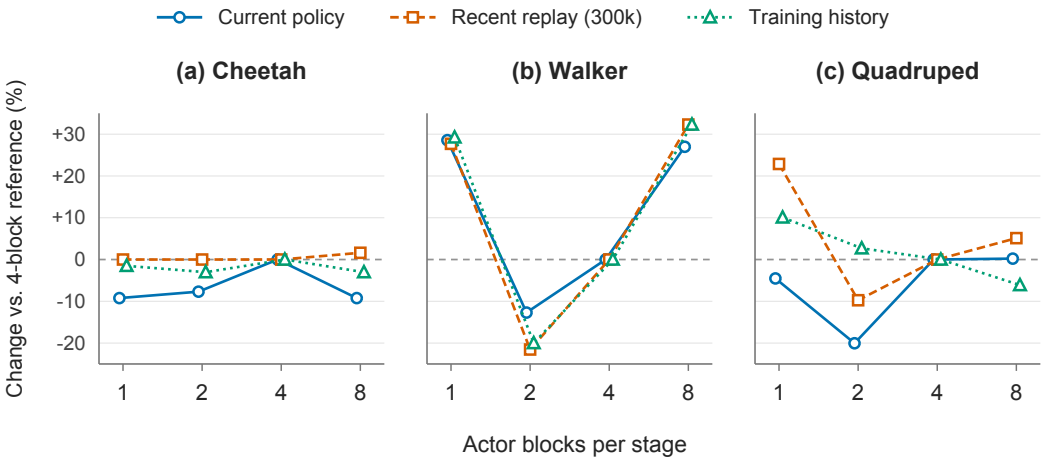


Figure 9: **Actor structure ablation** with four cascade stages and matched actor parameter counts. The axes show blocks per stage b and coverage change $100(C_b/C_4 - 1)$ (%) relative to four blocks, where C_b is coverage for each metric.

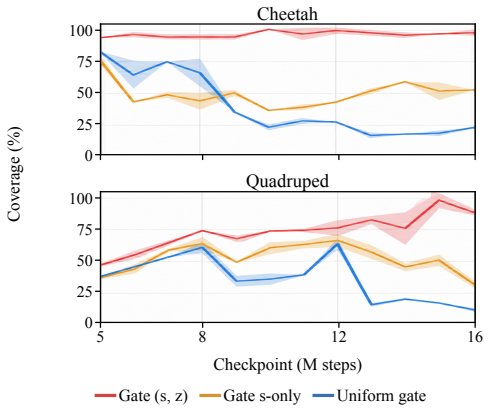


Figure 10: **Gate-input ablation.** Normalized coverage in Cheetah and Quadruped; lines average two evaluations and shading shows variance.

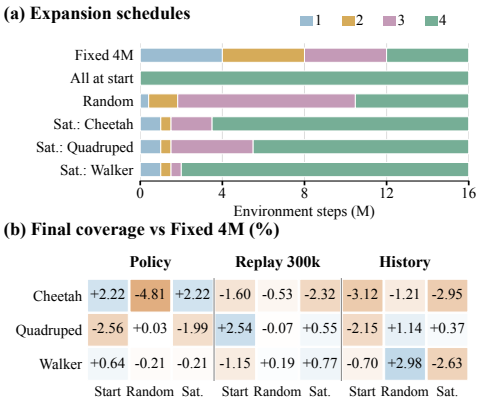


Figure 11: **Expansion timing ablation.** (a) Stage additions through 16M steps. (b) Final coverage change (%) from Fixed 4M, ordered as start, random, and saturation within each metric.

7 Conclusion

Existing unsupervised skill discovery typically optimizes exploration objectives within fixed policy architectures, limiting pretraining to trajectory distributions representable by the policy family. We propose EXPAND, which adds cascade stages during pretraining and recursively combines action distribution parameters through gates conditioned on state and skill, preserving the base method's intrinsic objective and skill interface. Our theory characterizes how policy changes affect state occupancy and gives conditions for local coverage gains from expansion and skill-conditioned gating. METRA experiments across seven environments and paired LSD and DIAYN comparisons show overall position coverage gains; parameter-matched controls support exploration gains from cascade organization at comparable capacity. EXPAND also improves METRA's overall zero-shot goal reaching and hierarchical control with frozen skills, supporting downstream skill reuse.

Gains vary with the base method and environment, and additional stages increase parameter and computational costs. We plan to organize and compose skills in a tree structure for dexterous manipulation. Our results support organizing and expanding policy capacity to improve exploration and skill reuse in unsupervised skill discovery while preserving the learning objective.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, including improving wording, grammar, and clarity, and to assist with literature retrieval and discovery, such as identifying potentially relevant related work. All AIassisted content was reviewed by the authors, and literature identified with AI assistance was checked against the corresponding original sources before inclusion. We take responsibility for the final content of this work, including all text, claims, citations, and artifacts produced with the aid of generative AI.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Chenjia Bai, Rushuai Yang, Qiaosheng Zhang, Kang Xu, Yi Chen, Ting Xiao, and Xuelong Li. Constrained ensemble exploration for unsupervised skill discovery. In *International Conference on Machine Learning (ICML)*, 2024.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

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)*, 2018a.

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*, 2018b. URL https://arxiv.org/abs/1812.05905.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Jiaheng Hu, Zizhao Wang, Peter Stone, and Roberto Martín-Martín. Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Zilin Kang, Chenyuan Hu, Yu Luo, Zhecheng Yuan, Ruijie Zheng, and Huazhe Xu. A forget-and-grow strategy for deep reinforcement learning scaling in continuous control. In *International Conference on Machine Learning (ICML)*, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised reinforcement learning benchmark. In *Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*, 2021.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R. Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. SimBa: Simplicity bias for scaling up parameters in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jiashun Liu, Johan Obando-Ceron, Aaron Courville, and Ling Pan. Neuroplastic expansion in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning (ICML)*, 2023.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations (ICLR)*, 2022.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations (ICLR)*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning (ICML)*, 2019.

Robin Schmöcker and Alexander Dockhorn. Cascade - A sequential ensemble method for continuous control tasks. *Reinforcement Learning Journal*, 6:399–411, 2025.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

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

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning (ICML)*, 2020.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018. URL https://arxiv.org/abs/1801.00690.

Hermann Thorisson. *Coupling, Stationarity, and Regeneration*. Springer, 2000.

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)*, 2012.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Rushuai Yang, Chenjia Bai, Hongyi Guo, Siyuan Li, Bin Zhao, Zhen Wang, Peng Liu, and Xuelong Li. Behavior contrastive learning for unsupervised skill discovery. In *International Conference on Machine Learning (ICML)*, 2023.

Chongyi Zheng, Jens Tuyls, Joanne Peng, and Benjamin Eysenbach. Can a MISL fly? Analysis and ingredients for mutual information skill learning. In *International Conference on Learning Representations (ICLR)*, 2025.

A Extended related work

Unsupervised exploration and skill discovery. Unsupervised pretraining seeks broad state coverage through active exploration or organizes reusable skills with latent variables (Laskin et al., 2021). Active exploration uses pseudocounts, prediction error, model disagreement, or information gain to guide state visitation (Bellemare et al., 2016; Pathak et al., 2017; 2019; Houthooft et al., 2016). APT constructs intrinsic rewards from self-supervised representations (Liu & Abbeel, 2021); we use it as a coverage reference without a skill interface. Skill discovery conditions policies on a latent variable z. DIAYN and DADS optimize mutual information (Eysenbach et al., 2019; Sharma et al., 2020). CIC uses contrastive intrinsic control, METRA learns metric representations, LSD imposes Lipschitz constraints, and CSF uses successor features (Laskin et al., 2022; Park et al., 2024; 2022; Zheng et al., 2025). BeCL contrasts behaviors at the trajectory level, while DUSDi factorizes skills (Yang et al., 2023; Hu et al., 2024). EXPAND is a framework for policy capacity expansion that changes only the skill-conditioned policy parameterization while preserving the base method's intrinsic objective and skill interface. We measure the module's incremental effect through paired comparisons of the same base method with and without EXPAND. METRA is the main instantiation; LSD and DIAYN test the same policy replacement under other skill-discovery objectives.

Policy capacity and multi-policy structures. Bias toward early experience and loss of network plasticity can hinder prolonged training in deep reinforcement learning (Nikishin et al., 2022; Lyle et al., 2023). Neuroplastic Expansion and Forget-and-Grow expand networks during training to mitigate these problems (Liu et al., 2025; Kang et al., 2025). SimBa uses normalization and residual connections to improve training of large actor-critic networks (Lee et al., 2025), while Cascade applies sequential ensembling to continuous control (Schmöcker & Dockhorn, 2025). EXPAND adapts Cascade's architecture to expand skill-conditioned policies while retaining the base method's intrinsic objective. CeSD organizes local exploration around state prototypes and combines skill occupancy constraints with value-function ensembles to learn skills (Bai et al., 2024). Its design involves the skill-learning signal, value architecture, and skill organization, so we include it as an algorithm-level reference. For a structural comparison with sequential cascades, we construct a Mixture of Policies (MoP) control that mixes the action probability densities of skill-conditioned experts in parallel.

B Finite-Horizon Perturbations and Cumulative Coverage

B.1 Sampling and counting conventions

Consider standard Borel state and action spaces. Let $\{B_1, \dots, B_N\}$ be a fixed finite measurable partition of the state space, with an integer cumulative-visit threshold $m_i \geq 1$ for each cell. For position coverage, B_i is the set of full states whose positions fall in the corresponding grid cell. A fixed-width grid over an unbounded domain need not satisfy the finite-partition condition.

Let k index a fresh trajectory and its skill (τ_k, z_k), where $s_{k,h}$ is the state at step h of that trajectory. Let $\mathcal{F}_k$ contain the history and training randomness available before collection, and let $\mathcal{L}(\cdot \mid \mathcal{F}_k)$ denote the conditional law given this information. The policy π_k is $\mathcal{F}_k$-measurable, remains fixed within that trajectory, and satisfies

$$\mathcal{L}((\tau_k, z_k) \mid \mathcal{F}_k) = q_{\pi_k}. \tag{B.1}$$

All collections use the same μ, P, $p(z)$, and H, but the policy may depend on the full history; trajectories need not be unconditionally independent. Coverage uses only newly collected data, and counts are never reset. Define

$$V_{k,i} = \sum_{h=0}^{H} \mathbf{1}\{s_{k,h} \in B_i\}, \qquad N_{k,i} = \sum_{\ell=0}^{k-1} V_{\ell,i},$$
$$K_k = \bigcup_{i:N_{k,i} \geq m_i} B_i, \qquad \Delta_k = \sum_{i=1}^{N} \mathbf{1}\{N_{k,i} < m_i \leq N_{k,i} + V_{k,i}\}. \tag{B.2}$$

Here $V_{k,i}$ and $N_{k,i}$ are, respectively, the visits to cell B_i in the current trajectory and the cumulative visits before collection; K_k is the region that has reached its threshold before collection; Δ_k is the number of cells that first reach their thresholds in the current collection; and $\mathbf{1}$ is the indicator function. Taking $m_i = 1$ gives first-visit coverage. Let $\eta_k = \bar{d}_{q_{\pi_k}}(\mathcal{S} \setminus K_k)$ be the time-averaged occupancy of the region that has not yet reached its threshold under the current policy. We use $d_{\mathrm{TV}}(\nu, \nu') = \sup_A |\nu(A) - \nu'(A)|$ for probability measures on the same measurable space, where A ranges over its measurable sets.

B.2 Finite-horizon perturbation bound and parameter consequence

Lemma B.1 (Policy perturbation and state occupancy)**.** *Two policies share the same μ, P, $p(z)$, and H. If*

$$\sup_{s,z} d_{\mathrm{TV}}(\pi(\cdot \mid s, z), \widetilde{\pi}(\cdot \mid s, z)) \le \varepsilon, \qquad \varepsilon \in [0, 1],$$

then their joint laws of skills and trajectories and time-averaged state occupancies satisfy

$$d_{\mathrm{TV}}(q_\pi, q_{\widetilde{\pi}}) \le H\varepsilon, \qquad d_{\mathrm{TV}}(\bar{d}_{q_\pi}, \bar{d}_{q_{\widetilde{\pi}}}) \le \frac{H}{2}\varepsilon. \tag{B.3}$$

Proof. Couple the two trajectories by sampling the same skill and initial state. While their histories have not diverged, use a maximal coupling for the actions (Thorisson, 2000), so that the conditional probability of different actions is at most ε; when the actions agree, use the common transition kernel to sample the same next state. The probability of divergence within the first h steps is at most $h\varepsilon$. The coupling inequality gives the total-variation bound for prefixes, and setting $h = H$ gives the trajectory bound. Let $d_{\pi,h}$ denote the state marginal at step h. Then

$$d_{\mathrm{TV}}(\bar{d}_{q_\pi}, \bar{d}_{q_{\widetilde{\pi}}}) \le \frac{1}{H+1} \sum_{h=0}^{H} d_{\mathrm{TV}}(d_{\pi,h}, d_{\widetilde{\pi},h}) \le \frac{\varepsilon}{H+1} \sum_{h=0}^{H} h = \frac{H}{2}\varepsilon.$$

The step-0 discrepancy is zero because the initial distributions agree. □

Replacing policy distance by parameter distance requires the additional assumption that $\Theta_0 \subseteq \mathbb{R}^p$, where p is the parameter dimension, $\|\cdot\|$ denotes the Euclidean norm, and a finite constant L_π exists such that

$$\sup_{s,z} d_{\mathrm{TV}}(\pi_\theta(\cdot \mid s, z), \pi_{\theta'}(\cdot \mid s, z)) \le L_\pi \|\theta - \theta'\|, \qquad \theta, \theta' \in \Theta_0. \tag{B.4}$$

For a finite feedforward network with Lipschitz activations, sufficient conditions are uniformly bounded state and skill features supplied to the network, parameters in a compact convex set Θ_0, and final Gaussian-policy standard deviations satisfying $0 < \underline{\sigma} \le \sigma_{\theta,d} \le \overline{\sigma} < \infty$ uniformly in every dimension. The mean and log standard deviation are then uniformly Lipschitz in the parameters; the Gaussian KL formula and Pinsker's inequality (Tsybakov, 2009, Lemma 2.5) yield Equation B.4, and the common $\tanh$ transformation cannot increase total variation. These conditions are used only for the parameter consequence. A finite position grid or finite parameter dimension alone does not ensure them, and boundedness on replay samples cannot replace uniform conditions.

For the realized policy sequence, fix K_k and define the future upper envelope of uncovered occupancy as

$$\Gamma_k(K_k) = \sup_{j \ge k} \bar{d}_{q_{\pi_j}}(\mathcal{S} \setminus K_k). \tag{B.5}$$

If $\pi_k = \pi_{\theta_k}$, all future parameters lie in Θ_0, and the remaining parameter drift $r_k = \sup_{j \ge k} \|\theta_j - \theta_k\|$ is finite, Lemma B.1 gives an occupancy-difference bound for each $j \ge k$. Taking the supremum yields

$$\Gamma_k(K_k) \le \eta_k + \frac{H}{2} L_\pi r_k. \tag{B.6}$$

This inequality holds pathwise and does not require parameter convergence. If, additionally, $\theta_k \to \theta_\infty$, then $r_k \to 0$. Both Γ_k and r_k depend on all future policies and need not be $\mathcal{F}_k$-measurable. They are not online expansion criteria, and small consecutive updates do not imply a small r_k.

B.3 Limits of cumulative counting and scope of interpretation

Lemma B.2 (Summability of uncovered occupancy)**.** *Under the sampling and counting conventions above, for any policy sequence adapted to the history,*

$$\mathbb{E}\left[\sum_{k=0}^{\infty}\eta_k\right] \le \sum_{i=1}^{N} m_i < \infty, \qquad \mathbb{E}[\Delta_k \mid \mathcal{F}_k] \le (H+1)\eta_k \longrightarrow 0 \quad \textit{almost surely}. \tag{B.7}$$

This result requires neither policy regularity nor parameter convergence nor a fixed architecture.

Proof. Let $U_{k,i} = \mathbf{1}\{N_{k,i} < m_i\}\mathbf{1}\{V_{k,i} > 0\}$. Each occurrence of $U_{k,i} = 1$ adds at least one visit, so pathwise $\sum_k U_{k,i} \le m_i$. Since $V_{k,i}/(H+1) \le \mathbf{1}\{V_{k,i} > 0\}$, the conditional sampling convention gives

$$\eta_k \le \sum_{i=1}^{N} \mathbb{E}[U_{k,i} \mid \mathcal{F}_k].$$

Taking expectations and interchanging the nonnegative sums gives the summability bound, and hence $\eta_k \to 0$ almost surely. Every cell that first reaches its threshold corresponds to at least one state observation outside K_k, so

$$\Delta_k \le \sum_{h=0}^{H} \mathbf{1}\{s_{k,h} \notin K_k\}.$$

Taking conditional expectations gives the second inequality and its limit. □

The finite partition also implies that K_k becomes constant after a finite random index k_0 on every sample path. When $j \ge k \ge k_0$, $K_j = K_k$; combined with Lemma B.2, this gives, almost surely,

$$\Gamma_k(K_k) = \sup_{j \ge k} \bar{d}_{q_{\pi_j}}(\mathcal{S} \setminus K_j) = \sup_{j \ge k} \eta_j \longrightarrow 0.$$

This limit likewise does not depend on parameter convergence.

The realized training path is not the whole policy family. Consider $H = 1$, a single skill, initial state s_0, and two actions that deterministically lead to s_L and s_R. Let π_θ move right with probability $\theta \in [0, 1]$, and use singleton cells with threshold one. If $\theta_k = 0$ for all k, then, for $k \ge 1$, $K_k = \{s_0, s_L\}$ and $\Gamma_k(K_k) = 0$. Yet π_1 in the same policy family reaches s_R with certainty, and its time-averaged occupancy of the uncovered region is $1/2$. Changing the intrinsic objective may still guide training to such a policy within the same family. Thus, the asymptotic limits above do not identify a capacity bottleneck or determine when a plateau occurs under a finite interaction budget.

C Coverage Conditions, Gate Benefits, and Implementation Details

C.1 Local boundary-crossing condition and distribution preservation

Proof of Proposition 2. The projection of a Gaussian distribution along n is Gaussian. Before expansion, its projected mean and variance are $n^\top \bar{\mu}$ and w; after expansion, they are $n^\top \bar{\mu} + (1-\lambda)\delta$ and Rw, respectively. Let Φ denote the standard normal distribution function. Then

$$p_0 = \Phi\left(-\frac{d}{\sqrt{w}}\right), \qquad p_+ = \Phi\left(\frac{(1-\lambda)\delta - d}{\sqrt{Rw}}\right). \tag{C.1}$$

Since Φ is strictly increasing, comparing the two arguments yields Equation 8. If entering an unvisited region in one step covers one cell, this probability difference also equals the difference in expected one-step coverage increments. □

Boundary approximation error. For a general transition kernel, fix a target region U, let $f_s(u) = P(U \mid s, \tanh u)$, and let $A = \{u : n^\top u \geq b\}$. For the pre-expansion and expanded distributions $j \in \{0, +\}$, let $\mathbb{E}_j$ denote expectation under the corresponding distribution, $p_j = \Pr_j(u \in A)$ its half-space hit probability, $\widetilde{p}_j = \mathbb{E}_j f_s(u)$ its true one-step arrival probability, and $e_j = \mathbb{E}_j|f_s(u) - \mathbf{1}_A(u)|$. The true arrival probabilities satisfy $|\widetilde{p}_j - p_j| \leq e_j$, and hence

$$\widetilde{p}_+ - \widetilde{p}_0 \geq p_+ - p_0 - e_+ - e_0. \tag{C.2}$$

The bound guarantees a positive true gain when the model gain exceeds $e_+ + e_0$. The e_j can be estimated by disagreement between half-space predictions and observed arrival labels on held-out transitions. Without validating the half-space model, this result provides only a local mechanistic prediction.

Distribution preservation and depth effect. Fix the existing prefix. The parameter recursion and the bijectivity of $\tanh$ imply that the full distribution remains unchanged if and only if

$$\mu_{\text{new}} = \bar{\mu}, \qquad v_{\text{new}} = \frac{1+\lambda}{1-\lambda}\bar{v}. \tag{C.3}$$

These equalities must hold for all (s, z), the new stage must represent the required functions, and the compensated variance must be within the allowed range. Fixed width and standard-deviation clipping can prevent exact preservation; adding a stage alone therefore does not imply policy-family inclusion.

If, at a given input, all stages have variance v_* and all gates equal λ, the recursion has the coordinatewise solution

$$v^{[m]} = \left[\frac{1-\lambda}{1+\lambda} + \frac{2\lambda}{1+\lambda}\lambda^{2(m-1)}\right] v_*. \tag{C.4}$$

This follows by substituting $v^{[1]} = v_*$ into the recursion. For $\lambda = 1/2$, the ratios are $1, 1/2, 3/8, \ldots$ and converge to $1/3$; they do not halve with every added stage. This conditional model suggests recording the aggregate variance in depth ablations. Action entropy also includes the Jacobian term of $\tanh$ (Haarnoja et al., 2018a), so variance alone does not determine changes in coverage or temperature gradients.

C.2 Proof of the value of skill-conditioned gating

Proof of Proposition 3. Here $\mathbb{E}_\nu$ denotes expectation under the fixed joint state–skill evaluation distribution, and $\mathbb{E}_s$ denotes expectation under its state marginal. The optimal values of the two selection rules are

$$V^*_{s,z} = \mathbb{E}_\nu \max(p_A, p_B), \qquad V^*_s = \mathbb{E}_s \max\{\mathbb{E}(p_A \mid s), \mathbb{E}(p_B \mid s)\}. \tag{C.5}$$

The identity $\max(a, b) = (a + b + |a - b|)/2$ yields Equation 9; nonnegativity follows from conditional Jensen's inequality. Let $D_+ = \max(D, 0)$ and $D_- = \max(-D, 0)$. Conditional on s, the benefit is the smaller of $\mathbb{E}(D_+ \mid s)$ and $\mathbb{E}(D_- \mid s)$, which gives the strict-gain condition. □

If the learned gates also choose between these two values, let their suboptimality relative to the respective optima be $r_{s,z}, r_s \geq 0$. Their realized difference equals the right-hand side of Equation 9, minus $r_{s,z}$ and plus r_s. Approximation error can eliminate the benefit of optimal selection, and joint retraining also changes stage outputs and the state distribution.

C.3 Initialization, training cycles, and expansion schedules

Algorithm 1 EXPAND unsupervised skill pretraining

Input: environment without extrinsic rewards, USD configuration and prior $p(z)$, interaction budget T, expansion cycles $\mathcal{E}_{\text{grow}}$, and maximum stage count M.

1 Initialize a single-stage policy, ψ, $\mathcal{D}$, and optimizers; set the current stage count to $m = 1$, $c = 0$, and cumulative interaction steps $t = 0$.
2 While $t < T$:
3 If $c \in \mathcal{E}_{\text{grow}}$ and $m < M$, initialize a new stage, register its parameters, and set $m \leftarrow m + 1$.
4 Run a USD collection-and-update cycle with $\pi_\theta^{[m]}$ in the original order. Collect at most $T - t$ steps and increment t by the actual count. Fix z within each episode and store it with the experience. Jointly update all stages and gates at each actor update.
5 Set $c \leftarrow c + 1$.
6 End while; freeze and return $\pi_\theta^{[m]}$.

In the METRA instantiation, when initialization from the preceding stage is enabled, the new stage copies compatible hidden layers, the mean head, and the log-standard-deviation head from the immediately preceding individual stage; it does not copy the entire existing prefix. The gate head is initialized separately with Xavier-normal weights (Glorot & Bengio, 2010) and zero bias. Zero bias does not imply a zero logit for every input. All stages retain the base actor's standard-deviation parameterization and clipping range. Expansion neither applies additional variance compensation nor resets the entropy temperature. New parameters are added to the same optimizer at the current actor learning rate and then jointly trained with existing stages.

The cycle c in Algorithm 1 denotes one configured collection-and-update cycle of the underlying method, rather than one trajectory in Appendix B.1. In METRA, each cycle first collects a batch of trajectories and then performs the configured training updates. If the collection batch in cycle c is $\mathcal{B}_c$ and trajectory τ contains $\ell(\tau)$ actual environment transitions, then

$$t_{c+1} = t_c + \sum_{\tau \in \mathcal{B}_c} \ell(\tau), \qquad t_0 = 0. \tag{C.6}$$

For parallel collection, the sum includes transitions from all trajectories. Only when every cycle collects n_{traj} complete trajectories of H steps, where n_{traj} is the number of trajectories per cycle, does $t_c = cn_{\text{traj}}H$; early termination and truncation at the end of the budget require actual step counts.

The preset cycle schedule is checked before collection in that cycle: a stage is added when $c \in \mathcal{E}_{\text{grow}}$ and the current number of stages is less than M. A fixed cycle interval g corresponds to positive integer multiples $g, 2g, \ldots$. When a schedule is described in interaction steps, the actual triggering values t_c should be reported; a fixed cycle interval equals a fixed interaction interval only when collection volume is constant in every cycle. Random and coverage-triggered schedules likewise require the actual expansion step counts. The underlying method's intrinsic objective, skill sampling, and collection and update order remain unchanged, and collection at the budget boundary is truncated by the remaining number of steps.

D Experimental Protocol

D.1 Environments, Coverage, and Downstream Evaluation

Pretraining uses OGBench AntMaze Giant and Large (Park et al., 2025), and five DeepMind Control Suite (DMC) environments (Tassa et al., 2018): Cheetah, Walker, Quadruped, Humanoid, and Fish. DMC uses the MuJoCo physics engine (Todorov et al., 2012). Coverage counts position grid cells visited by current-policy evaluation trajectories (Policy), stored in the replay buffer (Replay 300k), or visited throughout training (History).

All experiments use eight random seeds except the stage-contribution ablation, which compares prefixes and stage interventions at the same final checkpoint. Appendix D.7 describes this exception. The eight-seed setting applies to pretraining, parameter-matched comparisons, LSD and DIAYN comparisons, both downstream evaluations, and the actor-structure, gate-input, and expansion-timing ablations.

All environments provide $64 \times 64 \times 3$ RGB pixel observations. Pretraining episodes last 250 environment steps, and the replay buffer holds $300{,}000$ transitions. Table 2 lists the skill spaces; we sample one skill per pretraining episode and hold it fixed throughout that episode.

Table 2: Skill spaces used for pretraining.

Environment	Skill space
AntMaze Giant, AntMaze Large	2-dimensional continuous
Cheetah, Walker, Quadruped	16 discrete skills
Humanoid	2-dimensional continuous
Fish	3-dimensional continuous

Table 3 gives the pretraining defaults. Each epoch collects eight episodes, or 2,000 environment steps, before the configured gradient updates. All methods use SimBa backbones (Lee et al., 2025) for their policy and value networks, with four SimBa blocks by default and CNN encoders for pixel observations. Parameter-matched controls and actor-structure ablations vary the SimBa width or block count as specified in their respective experiments. The target-network coefficient $\rho = 0.995$ weights the previous target parameters: $\bar{\theta} \leftarrow \rho\bar{\theta} + (1-\rho)\theta$.

Table 3: Pretraining hyperparameters for METRA and EXPAND.

Hyperparameter	Value
Learning rate	10^{-4}
Optimizer	Adam (Kingma & Ba, 2015)
Episodes per epoch	8
Gradient steps per epoch	50 (AntMaze, Cheetah); 200 (Walker, Quadruped, Humanoid, Fish)
Minibatch size	256
Discount factor γ	0.99
Replay buffer capacity	3×10^5 transitions
Observation shape	$64 \times 64 \times 3$ RGB pixels
Pixel encoder	CNN
Policy and value network backbone	SimBa
Default number of SimBa blocks	4
Default hidden width	1,024
Target-network smoothing coefficient ρ	0.995
Entropy coefficient	Automatically tuned (Haarnoja et al., 2018b)
METRA ε	10^{-3}
METRA initial dual variable	30
Pretraining episode length	250 environment steps

Table 1 reports AUC (%) for the Policy, Replay 300k, and History position-coverage curves.

Appendix D.5 describes zero-shot goal reaching on Cheetah, Walker, and Quadruped. Hierarchical control covers three tasks each in Cheetah and Walker and compares methods with a skill interface. Appendix D.6 details how we freeze the low-level policy and train a high-level PPO policy to select z under the same downstream interaction budget.

D.2 Comparison Methods and Matched Settings

The paired EXPAND experiments fix the base method's intrinsic objective, environment interaction budget, and evaluation protocol, changing only the policy parameterization. Unless stated otherwise, EXPAND denotes the METRA instantiation (Park et al., 2024). We apply EXPAND to LSD (Park et al., 2022) and DIAYN (Eysenbach et al., 2019) on Cheetah, Quadruped, and Walker to test the same policy replacement under other skill-discovery objectives. CeSD (Bai et al., 2024), APT (Liu & Abbeel, 2021), and AptMetra represent alternative exploration designs. AptMetra retains a skill-conditioned policy and combines the METRA and APT intrinsic rewards after separate root-mean-square normalization; Appendix E.3 gives its definition.

The parameter-matched experiments use two single-policy controls: one widens both the SimBa policy and value networks, and the other increases their depth. MoP is a parallel multi-policy control

in which a learned router mixes the action distributions of jointly trained skill-conditioned experts. Appendix E.1 gives its definition.

D.3 Additional pretraining results

Figures 12 and 13 complement the History results in Figure 4. Both use nonlinear vertical axes with ticks in actual cell counts.

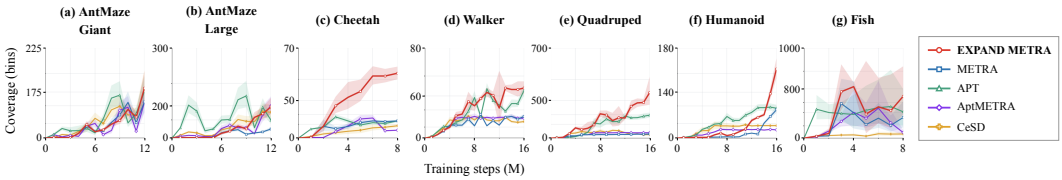


Figure 12: Pretraining Policy coverage: cells visited during checkpoint evaluation.

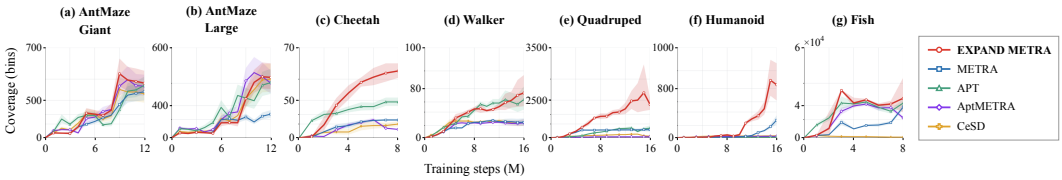


Figure 13: Pretraining Replay coverage: cells represented in the recent replay buffer.

D.4 Additional results with matched parameters

Figure 15 shows that EXPAND METRA has the highest final Replay 300k coverage in all five environments. Figure 14 shows that its final Policy coverage leads on Cheetah, Quadruped, and Humanoid, but trails Deep METRA on Walker and Widened METRA on Fish. Gains are more consistent for History and Replay 300k. All three plots use nonlinear vertical axes with ticks in actual cell counts.

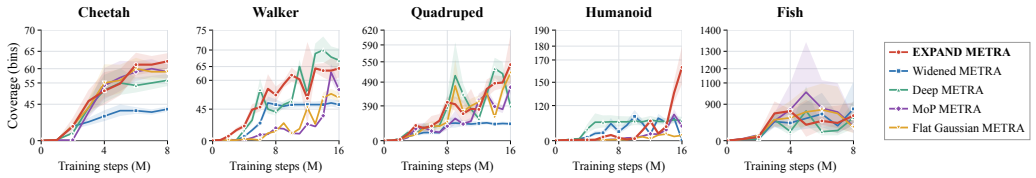


Figure 14: Policy coverage with matched parameter counts: cells visited during checkpoint evaluation.

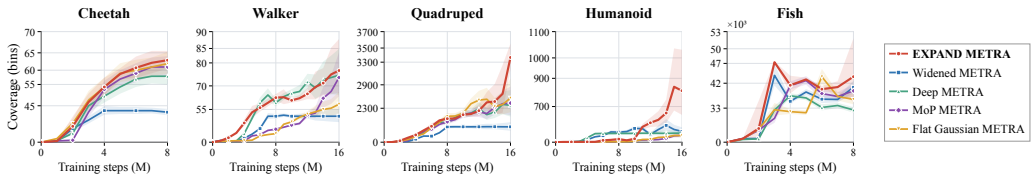


Figure 15: Replay 300k coverage with matched parameter counts: cells represented in the replay buffer.

D.5 Zero-shot goal reaching

Figure 6 reports frozen-policy evaluations on Cheetah, Walker, and Quadruped for 250 steps per goal using eight random seeds. Curves report mean success and final goal distance (metres), with rollout standard deviations. At 0M, success is anchored at zero and distance at the mean initial distance; neither is a checkpoint evaluation.

During APT pretraining, an additional goal-conditioned policy uses Contrastive RL (Eysenbach et al., 2022) with replay transitions and relabeled goals. It is frozen for evaluation and labeled APT (+CRL).

D.6 Hierarchical control with frozen skills

Figure 7 reports results on Cheetah Run Forward, Run Backward, and Flip, and Walker Stand, Walk, and Run. We freeze each pretrained skill policy and train high-level PPO (Schulman et al., 2017) to select z using task rewards for 500k environment steps. Each downstream episode lasts 200 environment steps. The high-level policy selects a skill every 50 steps, giving four skill selections per episode; the low-level policy executes the selected skill between decisions. EXPAND METRA uses four cascade stages with four blocks each. We use eight random seeds and report evaluation-episode means $\pm$ standard deviations.

We compare EXPAND METRA, METRA, AptMETRA, and CeSD. We exclude APT because it lacks a skill interface; we also exclude its additional CRL goal-conditioned policy used in zero-shot evaluation.

Table 4 lists the high-level PPO settings, including generalized advantage estimation (GAE) (Schulman et al., 2016). Episode lengths and skill intervals count low-level environment steps.

Table 4: Hyperparameters for the high-level PPO controller.

Hyperparameter	Value
Policy and value network backbone	SimBa
Number of SimBa blocks	4
Episodes per epoch	64
Gradient steps per episode	10
Minibatch size	256
Entropy coefficient	0.01
GAE λ	0.95
PPO clipping threshold ϵ	0.2
Episode length	200 environment steps
Skill selection interval	50 environment steps
Downstream interaction budget	5×10^5 environment steps
Random seeds	8

D.7 Prefix Coverage and Stage Interventions

The prefix-coverage experiment freezes one final checkpoint and constructs P_k from the first k cascade stages, where $k \in \{1, 2, 3, 4\}$. We run each prefix, recollect trajectories, and discretize the visited positions into cells. Let $\mathcal{V}_k$ denote the set of cells visited by P_k. The color of a cell is determined by the shortest prefix that reaches it:

$$\kappa(c) = \min\{k : c \in \mathcal{V}_k\}. \tag{D.1}$$

For Cheetah and Walker, each row corresponds to one discrete skill, and the horizontal axis is discretized into 1-m x cells. For Quadruped and Humanoid, we pool all skills and measure coverage in the x–y plane. Fish uses five intersecting planes with eight directions per plane, for a total of 40 continuous skills. We discretize visited states into 3-mm cells and show the 3D coverage together with its x–z, y–z, and x–y projections. Panel 8(e) shows the 3D structure, while the projections expose coverage on each coordinate plane.

For the stage intervention, we start from the full cascade, disable one target stage at a time, and recollect trajectories under the same evaluation protocol. Let $\mathcal{V}_{\text{full}}$ and $\mathcal{V}_{-i}$ denote the visitation sets of the full policy and the policy with stage i disabled. We define the change in coverage and the retention rate as

$$\Delta_i = |\mathcal{V}_{\text{full}}| - |\mathcal{V}_{-i}|, \qquad R_i = \frac{|\mathcal{V}_{\text{full}} \cap \mathcal{V}_{-i}|}{|\mathcal{V}_{\text{full}}|}. \tag{D.2}$$

The set difference $\mathcal{V}_{\text{full}} \setminus \mathcal{V}_{-i}$ locates regions affected by stage i.

D.8 Gate-Input Ablation

We train the cascade separately with (s, z) gates, s-only gates, or gates fixed to $1/m$ for the current number of stages. Cheetah and Quadruped checkpoints from 5M to 16M steps use equal evaluation budgets. Coverage is normalized by each environment's observed maximum; lines average two independent evaluations, and shading shows variance.

D.9 Expansion-Timing Ablation

All schedules reach four stages at 16M steps. Fixed 4M adds stages at 4M, 8M, and 12M; the initialization variant starts with all four. Random additions occur at 0.410M, 1.826M, and 10.478M. Saturation additions occur at $(1, 1.5, 3.5)$M in Cheetah, $(1, 1.5, 5.5)$M in Quadruped, and $(1, 1.5, 2)$M in Walker. Figure 11(b) reports changes from Fixed 4M.

Saturation criterion. We check cumulative History coverage every 500k environment steps using the same position grid throughout training. Let C_j denote the number of cells visited by step $t_j = j \times 500{,}000$, and let $\Delta C_j = C_j - C_{j-1}$ be the number of newly visited cells in the latest interval. We declare exploration saturated when the current increment falls below half of the preceding increment:

$$\Delta C_j < 0.5\, \Delta C_{j-1}, \qquad j \geq 2. \tag{D.3}$$

The first comparison is therefore at 1M steps. In the saturation-based schedule, this condition triggers one additional stage if fewer than four stages are active; training then continues within the same interaction budget.

E Baseline implementation details

Mixture of Policies and Flat Gaussian modify the policy parameterization, while AptMetra augments the intrinsic reward. All three retain METRA's skill-conditioned control interface and its original representation-learning objective (Park et al., 2024).

E.1 Mixture of Policies

MoP uses a mixture-of-experts architecture (Jacobs et al., 1991), replacing the single actor with K parallel experts and a learned router. Let $x(s, z)$ denote the actor input, formed by concatenating the observation features and skill z. Each expert receives the same input and defines a diagonal Gaussian followed by an elementwise $\tanh$ transformation:

$$u_k = \mu_k(x) + \sigma_k(x) \odot \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I), \qquad a_k = \tanh(u_k). \tag{E.1}$$

Writing its action density as $\pi_k(a \mid s, z)$, the router and mixture are

$$w_k(x) = \frac{\exp((Wx + b)_k / T)}{\sum_{j=1}^{K} \exp((Wx + b)_j / T)}, \tag{E.2}$$

$$\pi_{\mathrm{MoP}}(a \mid s, z) = \sum_{k=1}^{K} w_k(x) \pi_k(a \mid s, z). \tag{E.3}$$

Here $T > 0$ is the routing temperature. The router weights and biases are initialized to zero, giving uniform initial routing; the experts are independently initialized and all are present from the start of training. This construction mixes probability densities, rather than averaging actions, means, or variances. In contrast, Cascade recursively combines distribution parameters to produce a single tanh-transformed Gaussian.

The experts and router are jointly optimized with the SAC actor objective (Haarnoja et al., 2018a). For a minibatch of B replay inputs $x_i = x(s_i, z_i)$, one reparameterized action a_{ik} is sampled from every expert at each input. With $Q_{\min} = \min(Q_1, Q_2)$ and entropy coefficient α, the minibatch loss is

$$\widehat{\mathcal{L}}_{\mathrm{MoP}} = \frac{1}{B} \sum_{i=1}^{B} \sum_{k=1}^{K} w_k(x_i) \left[\alpha \log \pi_{\mathrm{MoP}}(a_{ik} \mid s_i, z_i) - Q_{\min}(s_i, z_i, a_{ik}) \right]. \tag{E.4}$$

The discrete expert index is summed out exactly; only the continuous actions are sampled. Gradients propagate through the routing weights, expert samples, and full mixture log density, which includes the tanh change-of-variables correction. The critic target uses an action sampled from the full mixture and its full mixture log density. All experts share the METRA representation, replay buffer, twin critics, and intrinsic reward; they are not separately trained METRA agents.

During stochastic interaction, an expert index is sampled from the router at each action decision, followed by an action from that expert. The deterministic action routine selects the expert with the largest routing weight and returns $\tanh(\mu_k(x))$. The implementation defaults are $K = 16$ and $T = 1$, also specified by the supplied MoP comparison launcher. Expert widths, depths, and hidden activations follow the base actor configuration. Unlike a single-actor update, Equation E.4 evaluates both critics on K actions per replay input, so parameter matching alone does not imply equal update cost.

E.2 Flat Gaussian Aggregation

Flat Gaussian replaces recursive aggregation with one softmax over the same m Gaussian stage outputs. Stage i predicts a mean $\mu_i(s, z)$ and diagonal standard deviations $\sigma_i(s, z)$; gates produce logits $g_i(s, z)$:

$$\begin{aligned} \ell_1 = 0, \quad \ell_i = g_i(s, z)\ (i > 1), \quad w_i = \frac{\exp(\ell_i)}{\sum_j \exp(\ell_j)}, \\ \mu_{\text{flat}} = \sum_i w_i \mu_i(s, z), \quad v_{\text{flat}} = \sum_i w_i^2 \sigma_i^2(s, z), \\ u \sim \mathcal{N}(\mu_{\text{flat}}, \operatorname{diag}(v_{\text{flat}})), \quad a = \tanh(u). \end{aligned} \tag{E.5}$$

The squared weights follow from summing conditionally independent Gaussian variables: given (s, z), $u = \sum_i w_i u_i$ with $u_i \sim \mathcal{N}(\mu_i, \operatorname{diag}(\sigma_i^2))$ has the moments above. Implementation samples once from the aggregate Gaussian and applies $\tanh$, yielding a single Tanh-Gaussian. "Flat" refers to computing all weights in one softmax.

E.3 AptMetra

AptMetra augments METRA with an APT-style exploration reward (Liu & Abbeel, 2021). It is enabled through METRA's mixed-reward option, rather than the standalone, unconditioned APT agent. Let r_i^{M} denote the original METRA intrinsic reward for a replay transition (s_i, a_i, s_i', z_i). A separate encoder f_ψ, independent of METRA's representation, supplies the features used to compute the APT reward.

The APT encoder is trained on current observations s_i using an asymmetric form of the InfoNCE loss (van den Oord et al., 2018). Let $h_i = p_\psi(f_\psi(\mathcal{A}_1(s_i)))$ and $\widetilde{h}_i = p_\psi(f_\psi(\mathcal{A}_2(s_i)))$, where p_ψ includes the projection head and $\mathcal{A}_1, \mathcal{A}_2$ are two views of the same observation. With stop-gradient operator sg and contrastive temperature τ_c, the loss is

$$\mathcal{L}_{\text{APT-rep}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(h_i^\top \operatorname{sg}(\widetilde{h}_i)/\tau_c)}{\sum_{j=1}^{B} \exp(h_i^\top \operatorname{sg}(\widetilde{h}_j)/\tau_c)}. \tag{E.6}$$

The views are identity transformations for state observations and independent random shifts for pixel observations. Both branches use the same encoder and projector; only the second branch is detached, with no separate moving-average target encoder.

The intrinsic reward uses the next observations in the sampled replay minibatch. Set $g_i = f_\psi(\mathcal{A}(s_i'))$ and let $\mathcal{N}_k(i)$ contain the k nearest representations to g_i within that minibatch, including g_i itself with zero distance. Under the default distance settings, the reward is

$$r_i^{\text{A}} = \log\left(1 + \frac{1}{k} \sum_{j \in \mathcal{N}_k(i)} \|g_i - g_j\|_2\right). \tag{E.7}$$

Thus the neighbors are drawn from the current minibatch, not a separate memory bank or the entire replay buffer. The code averages Euclidean distances before taking $\log(1 + \cdot)$; its default reward

neither raises distances to the representation dimension nor divides them by a running distance mean. These details specify the implemented APT-style reward.

Before mixing, the two reward components are normalized separately by their running root-mean-square scales. For $c \in \{\mathrm{M}, \mathrm{A}\}$, let $\widehat{\mathbb{E}}_t[(r^c)^2]$ denote the cumulative second moment over component rewards processed through update t, including the current minibatch. The mixed reward is

$$\bar{r}_i^c = \frac{r_i^c}{\sqrt{\widehat{\mathbb{E}}_t[(r^c)^2] + \varepsilon}}, \qquad c \in \{\mathrm{M}, \mathrm{A}\}, \tag{E.8}$$

$$r_i^{\mathrm{mix}} = \beta \bar{r}_i^{\mathrm{M}} + (1 - \beta)\bar{r}_i^{\mathrm{A}}. \tag{E.9}$$

No mean is subtracted, so normalization preserves each component's sign. The mixing coefficient β corresponds to `apt_reward_lambda` and weights the METRA reward, not the APT reward. Its default value, also used in the supplied mixed-reward comparison launcher, is 0.5.

For each pretraining minibatch, the METRA representation is first updated with its original objective, and the independent APT encoder and projector are updated using Equation E.6. Rewards are then recomputed, normalized, and mixed before the SAC updates. The mixed reward is detached, so actor–critic losses do not backpropagate into either representation through reward computation.

The APT module's configuration defaults are representation dimension 50, $k = 12$, $\tau_c = 1$, and $\varepsilon = 10^{-8}$. For state inputs, its encoder uses a 256-unit ReLU hidden layer followed by a 50-dimensional output with layer normalization (Ba et al., 2016) and tanh; the projection head applies ReLU to the representation, then a 128-unit ReLU hidden layer and a 64-dimensional layer-normalized output. The encoder and projector use a separate Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-3}. These are implementation defaults, not a claim that every experiment uses an identical configuration.