MAINTAIN YOUR RANKS: ENABLING CONTINUAL LEARNING IN RL AGENTS WITH GROWING AND PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The learning capacity of neural networks in deep reinforcement learning (RL) is essential for extracting task-specific representations, thereby facilitating learning. However, networks' capacity can be lost during training with standard gradient optimizers; timely intervention is needed to provide new capacity or release the underutilized capacity to cope with the non-stationary data distribution in RL. Despite the evolving data distribution, common RL practices overly commit to static network architectures, which can suffer from plasticity loss in the sense of lacking adaptability in learning, an inherent trait that can be captured by their collapsing effective rank (e-rank). We argue that maintaining a sufficient e-rank of the neural networks' layers is a necessary precondition for plasticity. To mitigate plasticity loss, we suggest a dynamic adaptation of the architecture, aiming to maintain the e-rank, by growing and pruning the neurons of a layer. We demonstrate that RL agents with adaptively growing-and-pruning critics can continually learn in both standard continuous control RL and continual learning for sequential RL tasks with superior performance.

1 INTRODUCTION

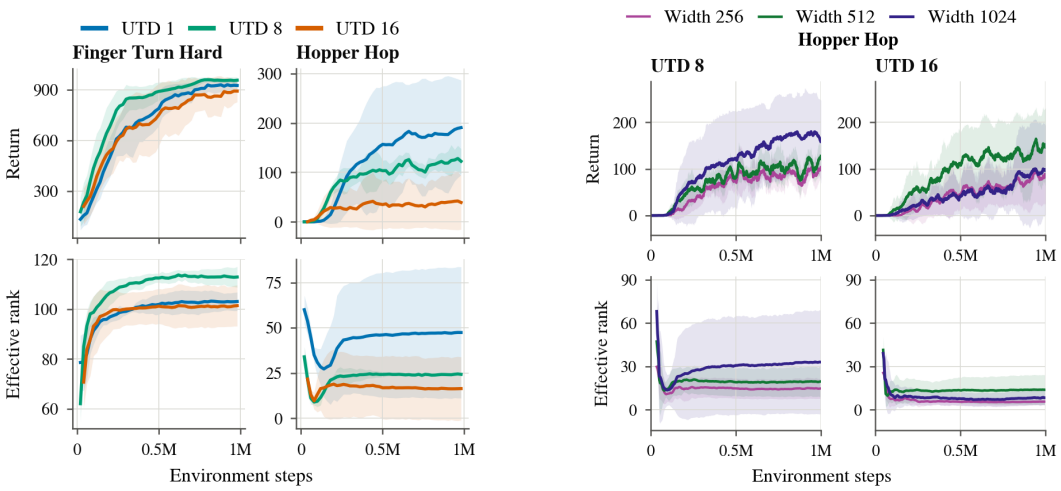
Continual learning in reinforcement learning (RL) is a setting in which agents never stop learning and must constantly adapt to ever-changing objectives and environments, and thus to continuously changing data distributions. Common RL practices rely on neural networks with a static structure and fixed capacity. Identifying a suitable architecture requires an extensive search or domain knowledge (Parker-Holder et al., 2022; Miao et al., 2022). Meanwhile, even when tuned, a fixed architecture is potentially ill-suited to shifting data distributions. This inability to keep learning from new data is called *loss of plasticity* (Lyle et al., 2023; Dohare et al., 2024).

In episodic RL, this loss of plasticity is often related to the primacy bias (Nikishin et al., 2022). An RL agent starts with no knowledge and learns from experience it collects itself, using a policy or value function whose network is initialized at random. Consequently, early experiences tend to be less task-specific. A network that fits this initial data too closely commits its representational capacity to a suboptimal distribution, and thus potentially suffers from low sample efficiency in learning and decreases final task performance. In sequential or continual RL tasks (Abel et al., 2023), this problem is exacerbated because the agent must adapt to entirely new tasks and dynamics. Both setups demand maintaining plasticity, in the sense of an adaptive capability to handle changing data distributions during continual training. Static networks without any plasticity intervention can be ill-suited for these setups, as shown in Figure 1.

A natural way to address the capacity gap is to adopt a dynamic architecture that changes during training. While various attempts at architectural adaptation have successfully improved performance on standard RL tasks (Nikishin et al., 2023; Kang et al., 2025) and sequential RL setups (Liu et al., 2025), their adaptability relies on rigid and periodic schedules that do not account for the neural network's actual representational capacity and expressiveness gap.

We propose an automated growing and pruning mechanism that is driven by the network's intrinsic representation dimensionality, quantified by the layer-wise effective rank (Roy & Vetterli, 2007; Kumar et al., 2020; Dohare et al., 2024). For example, Kumar et al. (2020) demonstrated that the

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



(a) (b)

Figure 1: Performance and the penultimate layer's effective rank of the Soft Actor-Critic (SAC) in DMC environments with different update-to-data (UTD) ratios and critic widths. Solid lines and shaded areas show the mean and 1× standard deviation over five random seeds. (a) As the update-to-data (UTD) ratio increases, meaning more plasticity loss as discussed in Nikishin et al. (2022), both performance and effective rank decrease. (b) A static-width critic can struggle with certain UTD ratios. Similarly, performance loss happens when the layer shows a suboptimal effective rank.

health of the learning process is intrinsically tied to the dimensionality of these feature representations. The effective rank measures how many singular values of a feature matrix are appreciably large, indicating the number of independent features a layer represents. When a layer's effective rank saturates, it has exhausted its representational capacity and requires more parameters (i.e., neurons) to represent richer features (growing). Conversely, when a layer falls short of the effective rank, it retains redundant neurons that duplicate existing directions. Such behavior is encouraged in deep neural networks to promote regularization and robustness (Srivastava et al., 2014; Morcos et al., 2018). However, in a continual learning setup, over-committing to specific directions can be harmful (Figure 1); thus, removing redundant neurons consolidates the network without substantially altering the learned representations (pruning). By interleaving these interventions, the network continuously adapts its structural capacity to the complexity of the incoming data.

Contributions. (i) We propose a data-driven grow-and-prune method for continual RL that actively regulates network capacity. Unlike schedule-based approaches, our method directly uses the layer-wise post-activation effective rank to dynamically trigger network expansion when capacity saturates and execute pruning when neurons become redundant. (ii) We provide evidence that per-layer effective rank acts as a reliable diagnostic for learning health in RL: a collapsing effective rank leads to representation collapse and training failure, while maintaining a high effective rank is a prerequisite for sustained learning. (iii) We demonstrate that this method autonomously scales network capacity and therefore copes with the shifting data distribution, thereby mitigating plasticity loss. In sequential RL, it learns new tasks faster than baselines and avoids catastrophic forgetting on 4 of 5 tasks. In standard RL, it reaches at least 95% of the best baseline's return in 15 of 18 setups.

2 Related Work

This section reviews work on plasticity loss and established measures, adaptive architectures as a mitigation approach, and the use of effective rank as a measure of dimensionality.

2.1 Loss of plasticity in RL

Unlike supervised learning on stationary datasets, deep RL operates in a non-stationary setting with shifting data distributions and moving bootstrapping targets. In this case, neural networks can ex-

hibit plasticity loss, losing the ability to learn from new data, leading to early performance plateaus and catastrophic forgetting (Nikishin et al., 2022; Dohare et al., 2024; Lyle et al., 2023). To diagnose this phenomenon, various measures have been proposed, including dormant neuron ratio (Sokar et al., 2023), the sharpness of the loss curvature (Lewandowski et al., 2023), the effective learning rate (Lyle et al., 2024), and the effective rank of feature representations (Dohare et al., 2024). However, while these measures effectively or complementarily quantify plasticity loss, they are often used as post hoc analytical tools rather than as an active trigger for architectural changes to bridge the network capacity gap during training.

2.2 Adaptive neural architecture in RL

To combat capacity saturation and plasticity loss in RL, recent literature explores adaptive architectures that alter their structure during training time.

Growing Expanding a network's capacity can mitigate primacy bias by allocating new parameters to capture task dynamics while allowing early representations to remain intact. However, determining *when* and *how* to grow typically introduces complex scheduling decisions. Nikishin et al. (2023) propose Plasticity Injection (PI), which freezes the Q-network head and adds parallel *trainable* parameters and frozen copies in a function-preserving manner. Similarly, GrowNN (Fehring et al., 2025) incrementally adds new function-preserving layers to Proximal Policy Optimization (PPO) agents (Schulman et al., 2017). Forget-and-Grow (Kang et al., 2025), based on the periodic reset proposed by Nikishin et al. (2022), adds new layers to the critic of Soft Actor-Critic (SAC) while decaying the replay of older data in the replay buffer. However, their periodic growth schedule coincides with their reset schedule, resulting in retraining new networks from scratch at each growth event. Crucially, these methods rely on *periodic* growth schedules that ignore the network's actual expressive gap and untimely growth, either prematurely (stronger primacy bias) or too late (after plasticity is lost), can result in suboptimal adaptation (see Appendix D).

Pruning To prevent unbounded parameter growth and eliminate redundant or underutilized capacity, several methods interleave growing with pruning. Neuroplastic Expansion (Liu et al., 2025) guides growth by the gradient magnitude on candidate new, randomly initialized, connections following a cosine annealing schedule, while pruning a subset of dormant neurons to enforce an upper bound on the layer size. Jia & Zhou (2026) propose SMGrNN, which uses statistics on activations and weight updates to trigger growth, alongside weight-based pruning that suppresses small-magnitude weights and truncates connections that lack recent updates. Similarly, Kong & Sutton (2026) balance the addition of new connections with the removal of low-magnitude weights to maintain a sparse topology. We argue that reliance on weight magnitude and update frequency for pruning can destroy consolidated neurons and hurt stable features. Further, pruning based solely on weight dormancy ignores the layer's structural geometry; a network with zero dormant neurons can still suffer plasticity loss if all active neurons are highly correlated and span a low-dimensional space.

2.3 Effective Rank

Effective rank (e-rank) (Roy & Vetterli, 2007) was originally proposed as a continual, entropy-based surrogate of the matrix rank to measure the information dimension in signals. In the context of neural networks, it has been repurposed to quantify the dimensionality of network representations (Dohare et al., 2024). Relatedly, stable rank measures the ratio of total matrix energy to the energy of its strongest single direction in the spectrum and has been used for similar diagnostic purposes (Kumar et al., 2020; Hussing et al., 2024), particularly applied to the features of the penultimate layer. A parallel line of work applies e-rank to the gradient direction to quantify gradient diversity (Lewandowski et al., 2025; Luo et al., 2026; Prakash et al., 2025). While Dohare et al. (2024) demonstrate e-rank as an effective diagnostic indicator of plasticity loss, we directly use the saturation and loss of the layer-wise e-rank to design our growing and pruning mechanisms to mitigate plasticity loss.

3 Effective-Rank-Based Dynamic Architecture

Motivated by the goal of maintaining representational capacity, we propose a growing-and-pruning technique based on the effective rank (e-rank) of post-activations. In particular, we dynamically

expand the layer width with newly initialized weights (**growing**) to provide redundant capacity for developing richer representations, and we subsequently remove redundant neurons that contribute (almost) trivial or duplicated variance (**pruning**) to eliminate collapsed feature dimensions. Combining them maintains an expressive yet frugal representation, enabling the RL agent to continuously develop new features and discard obsolete ones without catastrophic collapse.

3.1 Effective Rank

The effective rank measures how uniformly a layer's activations span its available distinct directions over a batch of inputs. A rich representation that spreads equally across all directions would have an effective rank equal to the number of neurons. Conversely, a collapsed representation whose features are concentrated in a single dominant direction would have an effective rank approaching 1.

Formally, following Roy & Vetterli (2007), consider a feature matrix $A \in \mathbb{R}^{m \times n}$ with singular values $\sigma_1, \sigma_2, \ldots, \sigma_q$, where $q = \min(m, n)$. Let $p_i = \sigma_i / \|\boldsymbol{\sigma}\|_1$, with $\boldsymbol{\sigma}$ being the vector of all the singular values and $\|\cdot\|_1$ the ℓ_1 norm. The *effective rank* of A, denoted by $\mathrm{erank}(A)$, is defined as

$$\mathrm{erank}(A) := \exp\{-\sum_{i=1}^{q} p_i \log(p_i)\} \tag{1}$$

While the standard matrix rank strictly counts the number of non-zero singular values, the effective rank is a continuous surrogate, satisfying $1 \leq \mathrm{erank}(A) \leq \mathrm{rank}(A)$ and explicitly measures the number of independent directions along which the matrix carries appreciable energy. It reaches its maximum when its spectrum is flat, i.e., all singular values have identical magnitude. In the context of deep RL, computing the effective rank of a neural network's layer measures the intrinsic representation dimensionality (Kumar et al., 2020; Dohare et al., 2024).

To evaluate this in an RL policy, we sample an independent probing batch of size $B \geq d$ from the replay buffer, producing a post-activation matrix $A \in \mathbb{R}^{d \times B}$ for a layer with d neurons. Importantly, we compute the effective rank on the mean-centered activations across the batch dimension. Nonlinearities like ReLU with a non-symmetric mapping can cause a mean shift that changes the spectrum, and the subsequent layer's weight and bias terms can arbitrarily transform this shift. When using effective rank to quantify *a layer's* representational dimension, we argue that the uncentered post-activation yields a distorted representational value, and that centering the signal removes the distortion and produces the usable feature dimensionality. We further elaborate on this in Appendix B.2 and illustrate it in Figure 6. In the following, we assume centered post-activations for the effective rank calculation unless specified.

Applying the effective rank to a neural network layer quantifies its intrinsic representational dimensionality, which is upper-bounded by the layer's width. Therefore, expanding the width provides the structural headroom to develop new features and skills. Fig. 1a empirically shows the relationship between the effective rank and the episodic return: across different update-to-data (UTD) ratios, a higher effective rank is accompanied by a higher episodic return. This observation motivates using the e-rank as a diagnostic of plasticity and as a trigger for adapting the network's capacity. Conversely, the effective rank, which reflects the spectral uniformity of the activation signal, can degrade for several reasons. W.r.t. Eq. 1, the degradation can stem from: 1) *saturated neurons* that contribute only a constant signal and are therefore associated with zero singular values after centering[1]; 2) *strongly correlated neurons* whose activation signals are functionally linear dependent and therefore collapsing the features to a low-dimensional span[2]; 3) *dominating neurons* that skew the centered activation signal, distorting singular value distribution p_i despite preserving the rank fully.

3.2 Growing

Adding neurons to a layer raises the dimensional ceiling, allowing the layer to learn more features and thus eventually the policy to adapt to more skills. We trigger growth when a layer's effective

[1] Saturated neurons only occur with certain but general types of activation functions, such as ReLU (also known as *dormant neurons* when saturated (Sokar et al., 2023)), tanh, and SELU.

[2] We note that fully linearly correlated neurons will decrease the rank and therefore reduce the effective rank. Strongly linearly correlated neurons, e.g., as quantified in Sec.3.3, will enlarge the gap between the rank and the effective rank, and provide similar or even redundant information.

rank plateaus relative to its current width. Specifically, at every I-th environment step, we measure each layer's effective rank (erank(A)) together with the *occupancy* $\rho :=$ erank$(A)/d$ for a layer with d neurons, indicating the fraction of the layer's width that its representation actively utilizes.

To capture this utilization trend, we fit a least-squares slope $\partial\rho$ over the last w occupancy measurements and compare it to a growth threshold ϵ_g. Using occupancy allows the threshold to remain consistent across varying widths. If $\partial\rho \geq \epsilon_g$, the layer is still expanding its feature span and no intervention is needed. If occupancy tends to flatten, i.e., $\partial\rho \leq \epsilon_g$, the layer has saturated in utilization, and we add neurons. The window is cleared at each growth event; the measurement window is flushed, and the slope $\partial\rho$ is re-evaluated when the window is filled with w new measurements.

Crucially, growing does not guarantee improved utilization; a layer's features may have been sufficient for the current task. Therefore, to prevent redundant expansion, we implement a stopping rule based on the absolute rank rather than the occupancy. Let e^- denote the effective rank before a growth event. When the current occupancy plateaus again, we measure the new effective rank e^+, and we permit further growth only if the effective rank shows meaningful improvement: $\frac{e^+ - e^-}{\mu} > \Delta$, where $\mu = \frac{1}{2}(e^+ + e^-)$. Intuitively, the gain in effective rank between successive growth events, relative to its current level, must be large enough to count as an improvement. Note that this stopping rule relies on the effective rank rather than the occupancy, since after each growth, the new neurons may only partially contribute new features, potentially causing the occupancy to drop while the effective rank can increase. In general, saturation in occupancy means the layer has slowed developing new features given its current width and current collected experiences; therefore, growth can be motivated to further develop it. On the other hand, saturation in effective rank means that likely no more meaningful features will be developed for the current task; therefore, growth can be stopped. On sequential tasks, we reset this stopping criterion to allow for fast learning.

3.3 Pruning

Pruning reclaims underutilized capacity associated with a collapsing feature. In particular, we aim to remove neurons that provide only redundant information. Our pruning method consists of two steps: clustering similar neurons and removing them based on the magnitude of their activation.

First, we measure the cosine similarity between the normalized neuron activations. Consider a centered post-activation matrix $M \in \mathbb{R}^{d\times B}$; we denote its i-th row by $M_{i,:}$, where each row corresponds to a neuron's activation with standard deviation $s_i = \sqrt{\mathrm{Var}(M_{i,:})}$. Define the similarity matrix $L \in \mathbb{R}^{d\times d}$ with its element $L_{i,j} = |\langle u_i, u_j\rangle|$, where $u_i := M_{i,:}/||M_{i,:}||$ is the normalized neuron activation, with dissimilarity $D = 1 - L$. To identify similar neuron activations, we apply complete-linkage agglomerative clustering on D with cutoff $1-\tau$ to generate the cluster $\mathcal{C}(\tau)$, where $\tau \in [0, 1]$ is the similarity threshold. The clustering groups similar neurons based on the similarity threshold τ in the sense that $L_{i,j} \geq \tau$ for every pair inside a cluster.

Second, we use the signal magnitude to identify and prune redundant neurons in each cluster $C_k \in \mathcal{C}(\tau)$, where we explicitly target saturated neurons and linearly dependent neurons. If every neuron in a cluster C_k produces only a trivial standard deviation $s_i < \epsilon_p$ for all $i \in C_k$, with ϵ_p being a small constant, the cluster contains only saturated neurons, denoted by $C_{\text{saturated}}$, and thus is pruned. For each remaining $C_k \in \mathcal{C}(\tau)\backslash C_{\text{saturated}}$, we keep the single neuron with the highest importance E, quantified by the product of the activation's standard deviation s_i and the outgoing weight norm $||W^+_{:,i}||$, i.e., $E_i := s_i||W^+_{:,i}||$, and remove the other neurons in the cluster. This importance-based pruning will encourage preserving the neuron structural influence on the subsequent layer, thereby reducing the behavioral perturbation of pruning and maintaining stability.

4 Experiments

In this section, we evaluate the efficacy of our growing-and-pruning method in mitigating plasticity loss and sustaining competitive performance and sample efficiency in higher update-to-data (UTD) ratios. We implement our method within the Soft Actor-Critic (SAC) (Haarnoja et al., 2018) framework based on the CleanRL library (Huang et al., 2022). We compare against Vanilla SAC with layer normalization (LayerNorm), periodic resets (Nikishin et al., 2022), Plasticity Injection (PI) (Nikishin et al., 2023), and Neuroplastic Expansion (NE) (Liu et al., 2025). We use LayerNorm in all the

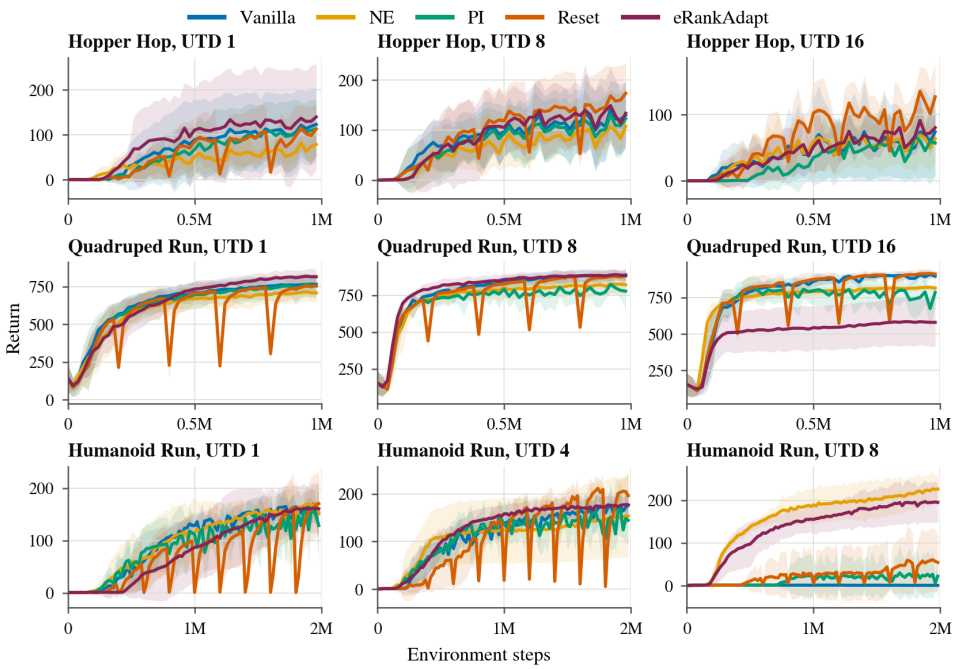


Figure 2: Performance of standard continuous control environments. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

baselines' critics to ensure results are directly comparable and to isolate the performance gain from LayerNorm alone (see Appendix D). Full hyperparameter details are reported in Appendix C.

4.1 Continuous Control

Setup We use standard continuous control from DeepMind Control Suite (Tassa et al., 2018). We focus on environments that have previously been shown to cause plasticity loss in RL agents (Nikishin et al., 2022). We conduct experiments up to a UTD ratio of 16, averaging results over 5 seeds.

Results We examine whether our growing-and-pruning intervention can sustain plasticity and achieve competitive performance in standard continuous RL tasks. As shown in Figure 2, our method's mean final return is within 5% of the best-performing baseline's in 5 of 9 setups and at or above it in 4 of them. We observe a single failure mode on `quadruped-run` at UTD 16, where our method underperforms relative to other baselines. The inspection of the network parameters reveals that the selected growth-stopping threshold is miscalibrated for this particular task, resulting in aggressive growth in early training, triggering the stopping criterion, and locking the critic at a fixed capacity for the rest of the training duration. We further evaluate the performance of the static critic network with different widths in the same setup in Appendix E. We observe that, in the individual environment-UTD ratio combination evaluated in this section, a well-tuned static width critic may achieve performance comparable to the aforementioned baselines. Moreover, Figure 8, Table 1 and Figure 17 (Appendix) confirm that our dynamic network performs on par with or surpasses the static net with a similar final size but trained from scratch in 5 of the 9 setups. Consequently, excluding the extreme `quadruped-run` at UTD 16 case, our dynamic method successfully eliminates the need for exhaustive architecture search without compromising final task performance.

4.2 Random Friction

Setup To evaluate our method's robustness to continuous data distribution shifts, we employ an environment wrapper that randomly resamples the sliding friction of every geometry in the same locomotion tasks in the previous subsection. Resampling is deferred to the next episode boundary, so friction is constant within an episode and every return is attributable to a single value. To prevent the

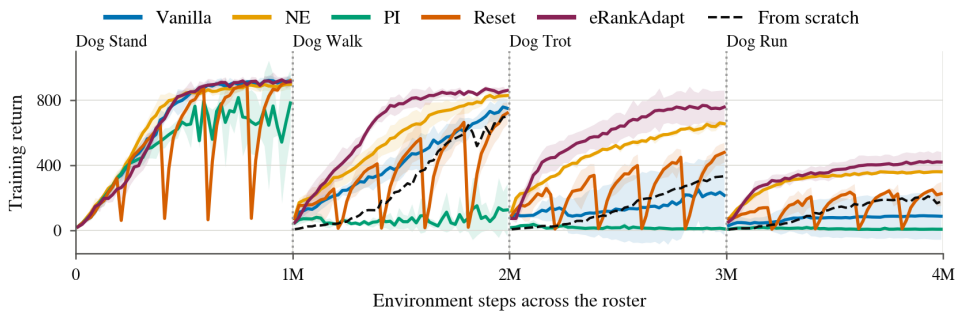


Figure 3: Sequential task performance in the `dog` environments. Results depict the return on the current task being trained. Solid lines and shaded areas show the mean and $1\times$ standard deviation, respectively.

network from anticipating periodic shifts, intervals between changes are drawn from an exponential distribution with mean $k = 100{,}000$ environment steps. Because the exponential is memoryless, this is equivalent to a change at the end of each episode with probability $1 - \exp(-L/k)$, where L is the episode length, approximately 1% per episode, or $\sim$10 changes per 1M steps. The new friction coefficient is drawn log-uniformly $[0.1, 2.0]$, i.e., bounds sit just outside the native environments'.

Results We ask whether our method can compete with well-established baselines and maintain plasticity when the data distribution constantly changes, inevitably forcing the policy to adapt to new information during training. We compare our method against the best-performing baseline in each of the 9 setups, which differ across setups. Our method's mean final return is within 5% of the best baseline's in 6 out of 9 setups (Figure 9) and at or above it in 3 of them. It falls short in two setups. On `hopper-hop` at UTD 16, it only exceeds Vanilla, NE, and PI. On `humanoid-run` at UTD 1, it has a high variance across seeds, with one seed failing to learn. Figure 10 shows the improvement in performance when we use critics of similar size to the final one achieved by our method and train it from scratch. Our method attains a higher IQM of final return in 7 of the 9 setups. The advantage is largest where Vanilla SAC fails to learn. The two setups where our method falls behind are both at UTD 1, most clearly `humanoid-run`. The benefit of adapting the architecture therefore grows with the replay ratio in which a fixed-capacity critic is prone to plasticity loss.

4.3 Sequential Tasks

Setup We construct a continual-learning setup in which we train the policy using a task from a specific environment and then change the task on a curriculum within the same embodiment after the agent has converged. The aim is to evaluate the agents' ability to adapt to a new reward function and data distribution converging on a previous task. The roster of tasks that we use is `humanoid-stand` $\rightarrow$ `humanoid-run` (1.5M steps/task), `quadruped-walk` $\rightarrow$ `quadruped-run` (1M steps/task), and `dog-stand` $\rightarrow$ `dog-walk` $\rightarrow$ `dog-trot` $\rightarrow$ `dog-run` (1M steps/task). All methods are task-aware: at task boundaries, the replay buffer is flushed, but the optimizer is retained. As a reference baseline, a static policy is trained from scratch on the new task; matching the data efficiency of such a policy indicates the preservation of plasticity and, therefore, the ability to learn continually. We sort the tasks in ascending order of difficulty because in our setting, the reward function for more difficult tasks usually includes that of the simpler ones, e.g., `humanoid-run` must be able to `stand` in order to `run`.

Results Figures 3, 11 and 12 show the learning curves following task transitions. In the `quadruped` and `dog` environments, our growing-and-pruning method exhibits remarkable data efficiency; it adapts to new tasks faster than the from-scratch reference and surpasses all baselines on each task, demonstrating that the network retains the capacity to rapidly acquire new skills.

In the `humanoid` environment (Figure 12), all baselines share similar learning curve profiles, with Neuroplastic Expansion being slightly above the rest; however, none of them undoubtedly outperforms Vanilla SAC (Welch's t-test, $p = 0.119$, 3 seeds).

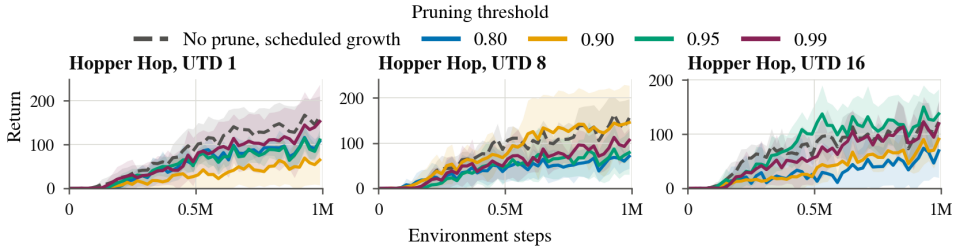


Figure 4: Ablation on similarity threshold τ in pruning with a fixed growing schedule. Solid lines and shaded areas show the mean and $1\times$ standard deviation, respectively. Also see Appendix F.

Among these task sequences, the most evident plasticity loss for the static network is in the four consecutive tasks on the `dog` environment, as shown in Figure 3: Vanilla SAC steadily loses its performance due to plasticity loss and failing to match from-scratch reference on the last two tasks, further emphasizing the pathology of static networks in the continual learning setup. Plasticity Injection (PI) collapses entirely on the second task, while Neuroplastic Expansion (NE) and Reset can remain competitive. However, our proposed method maintains plasticity across the whole task horizon and achieves exceptional episodic return and data efficiency. Figure 13 further demonstrates that the rapid adaptation of the proposed dynamic architecture does not induce catastrophic forgetting, retaining at least 98% of its mean evaluation return on previous tasks across 4 out of 5 tasks.

4.4 Ablation Study

To isolate the individual impacts of our algorithmic components, we ablate the pruning mechanism and the optimization regularization on the continuous control setup detailed in Section 4.1. We evaluate the sensitivity of the pruning mechanism's similarity threshold τ on `hopper-hop` across varying UTD ratios. We control for capacity additions by using a fixed-interval growth schedule. As shown in Figure 4, with a fixed-interval growth schedule and a mild pruning threshold $\tau \to 1$ only minimally degrades performance. More importantly, as detailed in Figures 18 to 20, our pruning method significantly reduces network width and increases the layer's occupancy without heavily compressing the effective rank. This confirms our pruning mechanism successfully targets and removes structural redundancy while fully preserving the intrinsic functional dimensionality of the learned representation.

We further investigate whether standard optimization regularization can enhance the representational benefits of our dynamics architecture by ablating the weight decay parameter λ in AdamW. Theoretically, weight decay explicitly penalizes the magnitudes of large weights and shrinks them. This uniform suppression of large weights could theoretically prevent dominant feature directions and, therefore, indirectly encourage a more uniform spectrum and improve the effective rank. However, our empirical results (Figure 5) contradict this hypothesis. We find that mild weight decay yields no significant performance improvements, whereas a large regularization strength may overly suppress weight development and significantly degrade performance. This inability of the standard weight-penalty mechanism to maintain spectral uniformity aligns with recent observations by Lewandowski et al. (2025). This, in turn, demonstrates that indirectly regulating weight magnitude is inadequate for maintaining representational health and highlights that the primary performance driver is from the structural interventions. As also noted in the future work, directly regularizing the spectrum may offer a more principled alternative to weight decay.

5 Limitations and Future Work

This paper focuses on adapting the width of the fully connected critic network in Soft Actor-Critic. In continuous control experiments, the proposed adaptive architecture successfully mitigates plasticity loss while keeping the actor network's capacity constant. In actor-critic RL, the actor's optimization is typically coupled with the critic's representation, and prior work has also shown that the optimal parameter ratio between the actor and the critic can vary across algorithms (Graesser et al., 2022). Therefore, simultaneously adapting the actor's architecture remains a highly promising di-

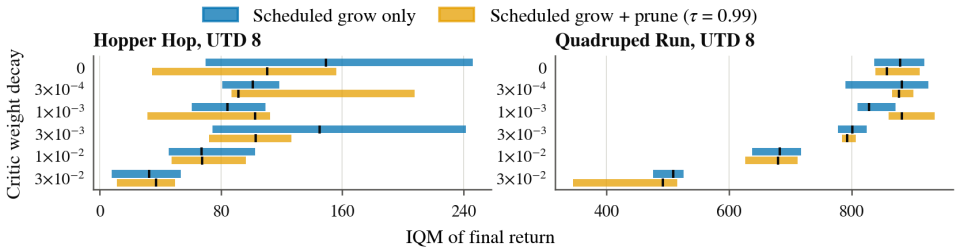


Figure 5: Ablation study on the weight decay coefficient λ in a fixed growing schedule with optional pruning with $\tau = 0.99$. We study $\lambda \in \{0, 3\times10^{-4}, 1\times10^{-3}, 3\times10^{-3}, 1\times10^{-2}, 3\times10^{-2}\}$. Each bar shows the IQM of the final mean performance, averaged over the last 10% of the environment steps, with a 95% confidence interval. Full results with effective rank in Appendix G.

rection. Moreover, extending our adaptive architecture to guide the network's depth growth, as well as extending this method to convolutional layers and residual networks for vision-based RL (Lee et al., 2025), presents an interesting direction.

A primary limitation of our current approach is its sensitivity to the effective-rank-based growth-trigger threshold, particularly the growth-triggering frequency, and the moving window size for growth-stopping. If these hyperparameters are strongly miscalibrated, the network may grow aggressively without sufficient gradient update steps. This can cause a slow increase in the effective rank and trigger the growth-stopping rule early, locking the network at a fixed capacity and terminating continual adaptation. We tuned the hyperparameters in the `hopper-hop` environment at UTD ratio 1; however, transferring across different UTD ratios and environments is not trivial. Our method would benefit from an adaptive threshold to further mitigate hyperparameter sensitivity.

Beyond structural interventions, an interesting future direction is to explicitly optimize for effective rank in gradient descent. A flat activation spectrum in the network can be encouraged through spectrum regularization with auxiliary loss terms (Lewandowski et al., 2025; Kumar et al., 2020), acting in a complementary manner with the proposed grow-and-prune architecture. We note that, while primarily evaluated in deep RL, this effective-rank-based dynamic architecture can be applied to broader continual learning settings, provided there is sufficient probing data to reliably estimate the effective rank of the representations.

6 Conclusion

We propose a data-driven architecture adaptation strategy for continual reinforcement learning. By utilizing the layer-wise post-activation effective rank to regulate network capacity, our method triggers structural expansion when representation capacity saturates and prunes neurons when feature redundancy occurs. Applied to the critics in Soft Actor-Critic, the proposed dynamic approach effectively overcomes plasticity loss inherent to static networks. Empirically, our method demonstrates strong performance across three continual learning scenarios. First, under high update-to-data ratios in standard RL benchmarks, it eliminates the need for manual architecture search by matching or surpassing the similar-width static network. Second, it rapidly adapts to continuous shifts in data distributions in random-friction environments. Third, in sequential RL tasks, our growing-and-pruning mechanism sustains outstanding plasticity, enabling rapid adaptation to new tasks while retaining performance on prior tasks without catastrophic forgetting. Furthermore, our experiments confirm that using the effective rank for active structural adaptation can surpass fixed-schedule interventions such as Plasticity Injection and Neuroplastic Expansion. While we identify one specific failure mode related to early capacity saturation from miscalibrated thresholds, our method promotes effective rank as a reliable driver for dynamic architectures. Overall, motivating architectural intentions based on intrinsic feature dimensionality makes a further step toward parameter-efficient, robust, continual RL agents.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for the implementation of methods and baselines and to provide feedback on research methodology and experimental setup. We have not used generative AI tools to formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, assist with translation, support qualitative and thematic data analysis, or interpret results, and generation of synthetic data sets, developing theoretical models or conceptual frameworks, cleaning and reformatting of datasets do not apply to this work. Additionally, we used generative AI tools for creating or editing software code, summarizing or analyzing existing literature, brainstorming, and editing to improve readability. We have reviewed all AI-assisted work. We have verified generated code and tested it for correctness, checked proposed literature for hallucinations and unsupported claims, and proofread all generated text. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

See the ICLR 2027 AI Policy for Authors for more details. This statement should not be more than 1 page.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

We have, to the best of our efforts, taken all actions to make this work as reproducible as possible. We include a detailed description of our method in Section 3. We provide an exhaustive list of experimental details and hyperparameters in Appendix C, including baselines.

REFERENCES

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

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

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

Lukas Fehring, Marius Lindauer, and Theresa Eimer. Growing with experience: Growing neural networks in deep reinforcement learning. *arXiv preprint arXiv:2506.11706*, 2025.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 7766–7792. PMLR, 2022.

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

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Marcel Hussing, Claas Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton. Dissecting deep RL with high update ratios: Combatting value divergence. *arXiv preprint arXiv:2403.05996*, 2024.

Yiyang Jia and Chengxu Zhou. Self-motivated growing neural network for adaptive architecture via local structural plasticity. *Neurocomputing*, pp. 134581, 2026.

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 *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Jeong Min Kong and Richard S Sutton. Plasticity of growing and elastic neural networks in online continual learning. *arXiv preprint arXiv:2608.01475*, 2026.

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

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter 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*, volume 2025, pp. 46816–46848, 2025.

Alex Lewandowski, Haruto Tanaka, Dale Schuurmans, and Marlos C Machado. Directions of curvature as an explanation for loss of plasticity. *arXiv preprint arXiv:2312.00246*, 2023.

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

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

Jiashun Liu, Johan S Obando Ceron, Aaron Courville, and Ling Pan. Neuroplastic expansion in deep reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 90056–90078, 2025.

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

Lirui Luo, Guoxi Zhang, Hongming Xu, Cong Fang, and Qing Li. Sphere: Mitigating the loss of spectral plasticity in mixture-of-experts for deep reinforcement learning. *arXiv preprint arXiv:2605.04712*, 2026.

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

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:106440–106473, 2024.

Yingjie Miao, Xingyou Song, John D Co-Reyes, Daiyi Peng, Summer Yue, Eugene Brevdo, and Aleksandra Faust. Differentiable architecture search for reinforcement learning. In *International Conference on Automated Machine Learning*, pp. 20–1. PMLR, 2022.

Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzciński, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36:37142–37159, 2023.

Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, et al. Automated reinforcement learning (AutoRL): A survey and open problems. *Journal of Artificial Intelligence Research*, 74: 517–568, 2022.

Arjun Prakash, Naicheng He, Kaicheng Guo, Saket Tiwari, Ruo Yu Tao, Tyrone Serapio, Amy Greenwald, and George Konidaris. Spectral collapse drives loss of plasticity in deep continual learning. *arXiv preprint arXiv:2509.22335*, 2025.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pp. 606–610. IEEE, 2007.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

A Appendix

B Reinforcement Learning and Soft Actor-Critic

Reinforcement Learning Formulation We model the reinforcement learning (RL) (Sutton et al., 1998) objective as a discounted Markov Decision Process (MDP), parameterized by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$. At each timestep, the agent observes a state $s \in \mathcal{S}$ and executes an action $a \in \mathcal{A}$ according to a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The environment subsequently transitions to a new state s' governed by the probability distribution $P(s'|s,a)$ and emits a scalar reward $r(s,a)$. The overarching objective is to discover an optimal policy π^* that maximizes the expected discounted cumulative return $J_t = \sum_{k=t+1}^{\infty} \gamma^{k-t-1} r_k(s,a)$, where γ serves as the discount factor. To estimate these returns, we utilize the state-value function $V_\pi(s) = \mathbb{E}_{\pi,P}[J_t | s_t = s]$ and the action-value function $Q_\pi(s,a) = \mathbb{E}_{\pi,P}[J_t | s_t = s, a_t = a]$.

Deep Actor-Critic Methods For continuous control tasks, we employ off-policy deep actor-critic architectures, specifically focusing on the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018). The SAC architecture relies on four core components: a parameterized policy network $\pi_\psi(a|s)$, a critic network $Q_\theta(s,a)$, a delayed target network $\bar{Q}_{\bar{\theta}}(s,a)$, and a replay buffer $\mathcal{D} = \{s_i, a_i, r_i, s_{i+1}\}_{i=1}^{N}$ used to store historical transitions.

With the exception of the target network, all components are optimized via gradient descent. To stabilize training, the target network parameters are updated at each time step using Polyak averaging, defined as $\bar{\theta} \leftarrow (1-\nu)\bar{\theta} + \nu\theta$, where ν controls the update rate. The policy (actor) and value (critic) networks are optimized using the following respective objectives:

$$\max_{\psi} \mathbb{E}_{\substack{s \sim \mathcal{D} \\ a \sim \pi_\psi(\cdot|s_i)}} \left[\min_{j \in \{1,2\}} Q_{\theta_j}(s,a) \right], \tag{2}$$

$$\min_{\theta} \left(Q_\theta(s,a) - \left(r + \gamma \mathbb{E}_{a' \sim \pi_\psi(\cdot|s_{i+1})} \left[\min_{j \in \{1,2\}} \bar{Q}_{\bar{\theta}_j}(s',a') \right] \right) \right)^2. \tag{3}$$

SAC distinguishes itself by augmenting its objective with a maximum entropy regularizer, $H(\pi_\psi(\cdot|s))$, which encourages exploration by maximizing the policy's entropy. To permit back-propagation through the expectation in Equation (2), the policy is drawn from a reparameterizable class of density functions.

B.1 Neural Networks and Their Layers

Consider a feedforward network of $K+1$ parameterized layers indexed by $k = 0, 1, \ldots, K$. A fully connected layer k holds d_k neurons and has a weight matrix $W_k \in \mathbb{R}^{d_k \times d_{k-1}}$, a bias $b_k \in \mathbb{R}^{d_k}$, an optional normalization layer ϕ_k with a per-neuron gain and shift, and an elementwise nonlinearity f_k. We call d_k the *width* of layer k. On a fixed probe batch of B samples, layer k maps the incoming features $X_{k-1} \in \mathbb{R}^{d_{k-1} \times B}$ to

$$Z_k := W_k X_{k-1} + b_k \mathbf{1}_B^\top, \qquad X_k := f_k\big(\phi_k(Z_k)\big) \in \mathbb{R}^{d_k \times B}, \tag{4}$$

where $\mathbf{1}_B$ is the all-ones vector, $Z_k \in \mathbb{R}^{d_k \times B}$ is the *pre-activation* and $X_k \in \mathbb{R}^{d_k \times B}$ the *post-activation*. We drop the subscript when referring to a general layer.

We note that we use post-activation X to estimate the effective rank, following Kumar et al. (2020); Dohare et al. (2024) rather than the pre-activation for two reasons. First, post-activation refers to the layer's output after the nonlinear activation function. Next, the pre-activation is a product of the incoming feature $X_{k-1} \in \mathbb{R}^{d_{k-1} \times B}$ and the layer weight $W \in \mathbb{R}^{d_k \times d_{k-1}}$ and therefore its rank is upper-bounded by the width of the previous layer d_{k-1}, which subsequently upper-bounds the effective rank since $1 \leq \mathrm{erank}(A) \leq \mathrm{rank}(A)$ as shown in Roy & Vetterli (2007). In other words, when using pre-activation, the information from the previous layer is included, which may therefore lead to an underestimation of the layer's actual feature dimensionality.

B.2 Centered Effective Rank

Activation functions are typically nonlinear. Passing through the activation signal does not guarantee that information will be retained or lost. In particular, activation functions like ReLU have a non-symmetric mapping, which biases the output distribution toward a particular direction. Following the practice in Hussing et al. (2024), we fit two Q-networks, as in standard SAC, to offline data collected with a random agent in the `hopper-hop` environment and investigate their centered and uncentered effective ranks. The result is presented in Figure 6. It can be observed that over the training horizon, the uncentered effective rank is always smaller than the centered effective rank. This gap arises in part from the nonlinear activation shifting the activation signal, concentrating energy in a particular direction, thereby reducing the uniformity of the activation signal and the effective rank. Since the subsequent layer may further shift the activation signal, and the bias term of the next layer may arbitrarily cancel out this gap, we argue that using the uncentered effective rank may underestimate the actual available representational dimension and therefore opt for using the centered effective rank to explicitly account for this offset. Centering refers to subtracting the per-unit means in the activation signal as follows.

$$\tilde{X} = X - \mathbf{1}_B \bar{x}^\top, \qquad p_i = \frac{\sigma_i(\tilde{X})}{\sum_j \sigma_j(\tilde{X})}, \qquad r = \exp\Big(-\sum_i p_i \log p_i \Big). \tag{5}$$

where $\bar{x} = \frac{1}{B} X^\top \mathbf{1}_B \in \mathbb{R}^d$ is the vector of per-unit means. We note that the operation of centering by subtracting the mean can reduce the rank and the effective rank by at most one.

B.3 Implementation Details

We initialize new neurons using LeCun normalization (Klambauer et al., 2017) to encourage diverse initial feature directions and mitigate their immediate pruning. Following common RL practices, we use the Adam optimizer with decoupled weights (Loshchilov & Hutter, 2017) and additionally carry the momentum over structural changes to accelerate learning for newly grown neurons. In Soft Actor-Critic (SAC) implementations, we allow each of the two critics to grow independently,

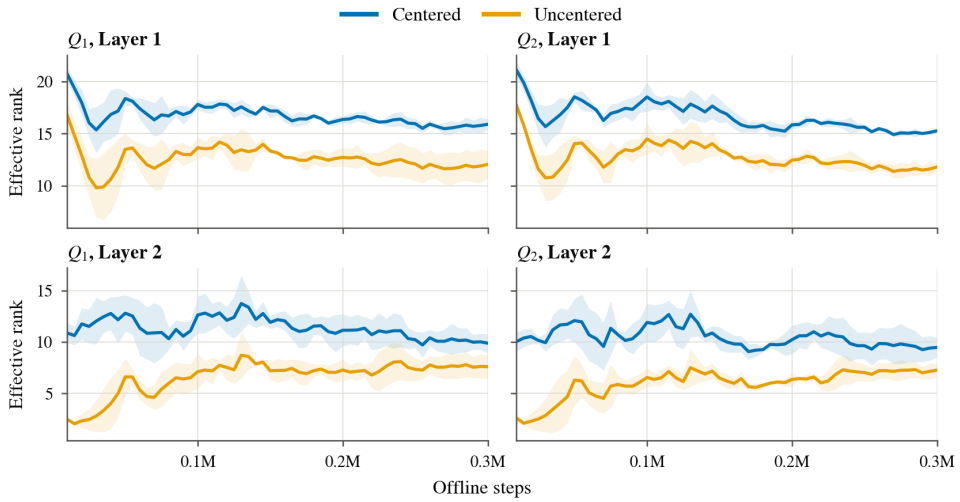


Figure 6: Centered and uncentered effective ranks of a 2-layer MLP with a fully connected layer and ReLU activation fitted to offline data collected in `hopper-hop`. Results are averaged over three random seeds. Solid lines and shaded areas show the mean and $1\times$ standard deviation, respectively.

as they regress on non-stationary bootstrapped targets and may lose plasticity at different rates. We only grow the hidden dimensions and keep the first layer's input dimension and the last layer's output dimension intact to preserve the network's task-specific nature. Our critics use layer normalization (LayerNorm) (Ba et al., 2016) for robustness and weight decay to mitigate the effective learning rate decay (Lyle et al., 2024). We also use a cosine annealing schedule with an adaptive horizon. After each growth step, the learning rate is reset to its initial value, and the horizon is set so that the schedule reaches its minimum at the total environment step budget. Cycles therefore become progressively shorter as growth events occur later in training. The actor is kept fixed in all cases. We ablate these interventions in Appendix D.

C Experimental setup and Hyperparameters

General setup All the experiments shown in this paper use Soft Actor-Critic (SAC) (Haarnoja et al., 2018) adapted from the CleanRL library (Huang et al., 2022). The actor is always a fixed 256×256 MLP. The baselines use 256×256 critics with LayerNorm. The optimizer is AdamW for both the actor and the critics, with learning rates of 3×10^{-4} and 1×10^{-3}, respectively, and a weight decay coefficient $\lambda = 1\times10^{-3}$. We use a batch size of 256, and the replay buffer holds 1M transitions, with 10% held out for fair diagnostics. The learning starts depending on the update-to-data ratio (UTD) using the formula $10K \times \sqrt{UTD}$. Results are reported using mean $\pm$ standard deviation unless specified otherwise.

Our method In our method, the critics always start with hidden dimensions 32×32, and each growth event adds up to 32 new neurons initialized using the LeCun normalization (Klambauer et al., 2017) and zero bias. Each of the two critics grows independently. We only grow the hidden dimensions and keep the first layer's input dimension and the last layer's output dimension intact. The critics' learning rate follows a cosine schedule with an adaptive horizon and weight decay 1×10^{-3}.

Growth Every I steps, each layer's effective rank is measured post-activation on a separate replay sample of 2048 transitions. A layer grows when the slope of its effective-rank / width ratio, taken over a window of $w = 30$ measurements, falls below $\epsilon_g = 0.0034$. Each layer of each critic is judged on its own. A target critic copies whatever happens to its online critic. Growth stops if the relative gain in rank per growth event is less than or equal to the threshold $\Delta = 0.0529$, judged over the last patience p events. Constant friction experiments use $p = 3, I = \lfloor \frac{850}{UTD} \rfloor$ and random friction experiments use $p = 1, I = 850$. Hyperparameters I, w, ϵ_g, Δ, and τ were tuned on `hopper-hop`

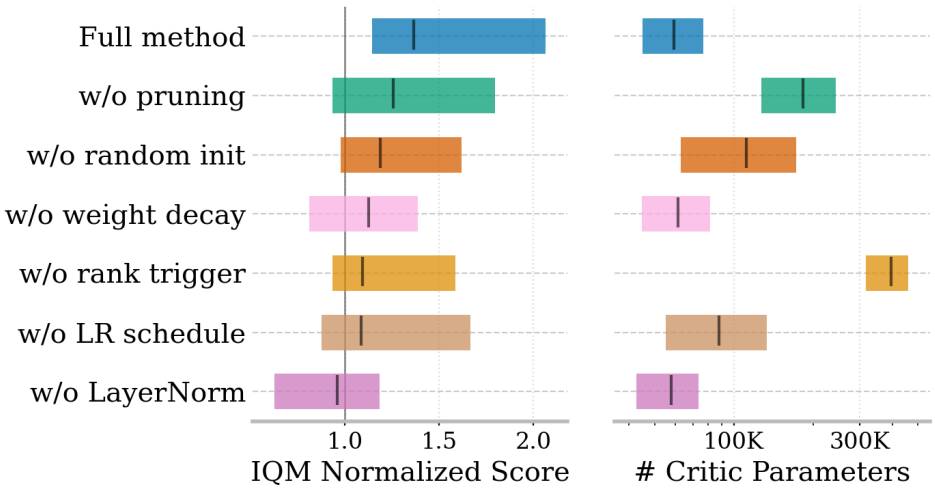


Figure 7: Ablation study on `hopper-hop`. Results on normalized interquartile mean of episodic return using $R(\pi) - R_{\text{random}}/R_{\text{SAC}} - R_{\text{random}}$ (left) and average number of parameters of the two critics (right).

UTD 1 using SMAC3 (Lindauer et al., 2022). The stopping signal for growth is permanent, with the only exception being the sequential setup, where it is reset when the task changes.

Pruning Pruning occurs immediately after each growth event. Units are clustered with complete linkage on the cosine similarity of their post-activation outputs, over 2048 samples, at a threshold of $\tau = 0.99$. One unit survives per cluster, and clusters of dead units are removed.

After each growth step, the learning rate is reset to its initial value, and the horizon is set so that the schedule reaches its minimum at the total environment step budget. Cycles therefore become progressively shorter as growth events occur later in training. The optimizer keeps its moments for the parameters that still exist.

Baselines Resets by Nikishin et al. (2022) are firing every $200K$ environment steps and are applied to the actor, critics, targets, the entropy temperature, and the optimizers. The replay buffer is kept. Plasticity Injection (Nikishin et al., 2023) is first applied at a quarter of the budget and then every $250K$ steps. According to the original implementation, it is applied only to the second-to-last layer (fc2). Neuroplastic Expansion (Liu et al., 2025) uses an initial sparsity of 25%; the topology is updated every $20K$ steps with growth rate $\alpha = 0.15$, a 20% prune ratio, and weight clipping $\kappa = 3$, also only on the second-to-last layer. We have found that experience review was harming performance, so we have disabled it.

Compute Experiments were run on NVIDIA V100 (16 GB and 32 GB), A100 (80 GB) and H100 GPUs, with each run on a single GPU.

D Further Ablation Study

To evaluate the contribution of each intervention, we conduct an ablation study on `hopper-hop` with an update-to-data ratio of 1. We utilize the leave-one-out (LLO) method and compare the normalized score with and without each intervention against the full setup. The results are shown in Figure 7. The normalized score is computed as $R'(\pi) = \frac{R(\pi) - R_{\text{random}}}{R_{\text{SAC}} - R_{\text{random}}}$, where R_{random} is the episodic return given by a random policy and R_{SAC} by Vanilla Soft Actor-Critic.

We notice that the use of LayerNorm has a major effect in stabilizing the results, making it highly important. It has been demonstrated by Nauman et al. (2024) that layer normalization is one of the most important elements when it comes to mitigating plasticity loss, especially when combined

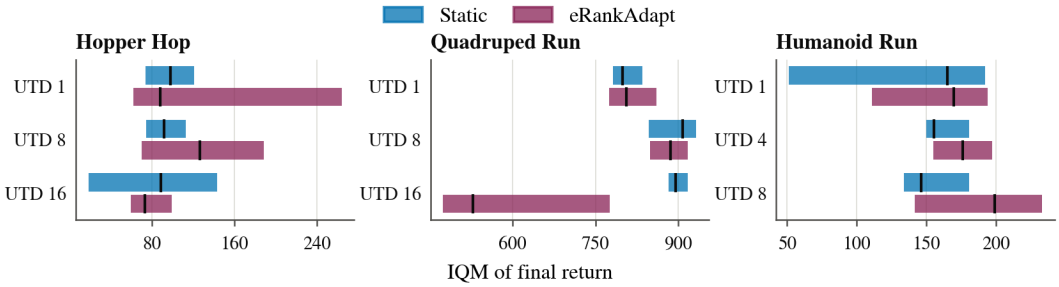


Figure 8: Interquartile mean of final episodic return for adaptive critics compared to training Vanilla SAC with similar hidden dimensions from scratch on standard static environments. Solid lines and shaded areas show the mean and $1\times$ standard deviation, respectively. Vanilla critics with hidden size 512 for `quadruped-run`, `humanoid-run` and 256 for `hopper-hop`.

with other interventions. This mostly holds in higher update-to-data ratios, but its effect can also be observed in UTD-2 in a much smaller magnitude. For this reason, all the baselines in this study use LayerNorm in the critic networks for a fair comparison and to showcase that the benefits come mostly from growing-and-pruning and not from LayerNorm. Layer normalization in this case enables the robustness of our method and should not be fully attributed for its performance.

Following this, we have the cosine scheduler that further improves performance. Using cosine annealing in a continual learning setup can be non-intuitive, but in our case it stabilizes the results, and its adaptive horizon and reset mechanism can account for the changes in the architecture.

We also see an expected improvement from using a trigger based on the effective rank to guide growth. The arm without a rank trigger uses a periodic schedule and grows stably every $50K$ environment steps. This number is not arbitrary but was tuned using SMAC3 (Lindauer et al., 2022) on `quadruped-run` and `hopper-hop`. We also notice that using a periodic schedule greatly increases the parameter size of the critics, as we cannot control how many parameters the layers actually need. This is consistent with the claim made in Section 2 and further supports the use of our e-rank trigger.

Not using weight decay further pushes the score below the base threshold. We attribute this to the effective learning decay (Lyle et al., 2024) and the benefit of suppressing the weight norm to maintain a high effective learning step.

Using a function-preserving initialization for new neurons also shifts the interquartile range further left, showcasing that introducing random directions is beneficial for e-rank and performance.

Finally, our pruning intervention provides a small benefit to episodic return, but the value is in the parameters. Not pruning redundant neurons results in critics nearly double the size. Thus, combining growing with pruning sets the pace for efficient agents that are not unnecessarily over-parameterized.

E Static Width

We conduct experiments on environments `quadruped-run`, `hopper-hop`, `humanoid-run` using SAC with different static widths $\in \{32, 64, 128, 256, 512, 1024\}$ for the critic and present the results on return, each hidden layer's effective rank, and occupancy (effective rank divided by the width) in Figure 14, Figure 15 and Figure 16, respectively. In general, SAC with a larger critic width performs better than one with a smaller width. We observe that width performance correlates with the layer's effective rank: higher effective ranks yield higher performance, whereas lower effective ranks fail to match higher ones. The deeper hidden layer (layer 2) shows a stronger correlation than the shallower layer (layer 1). Moreover, we find that the largest width does not always perform the best, as shown, e.g., in `hopper-hop` with UTD 16 and `humanoid-run` with UTD 4. Interestingly, we also observe that in these scenarios, the second hidden layer's effective rank is suboptimal, indicating feature collapse in the layer and leading to decreased performance. As for layer occu-

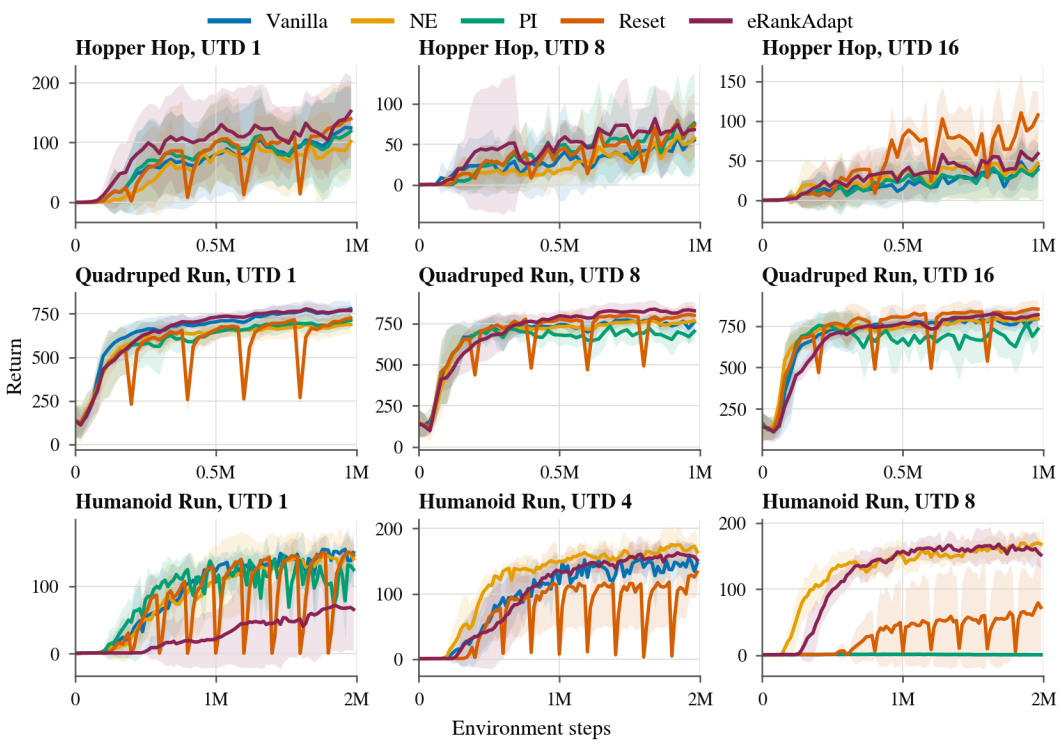


Figure 9: Episodic return of environments using random geom friction changes. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

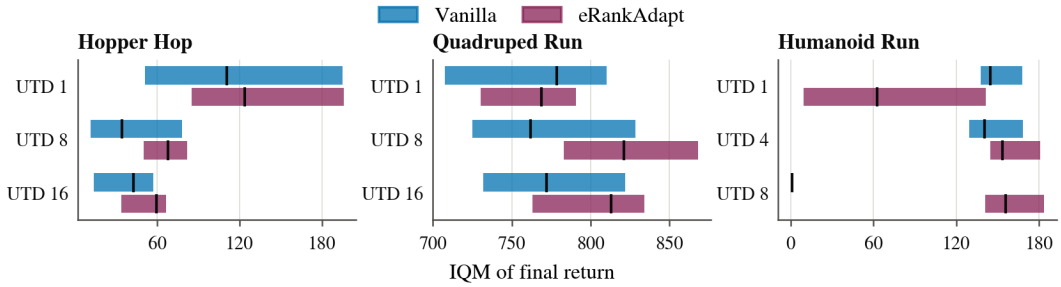


Figure 10: Interquartile mean of final episodic return using random geometry friction for adaptive critics compared to training Vanilla SAC with similar hidden dimensions from scratch. Mean and std over 5 seeds. Vanilla critics with hidden size 256.

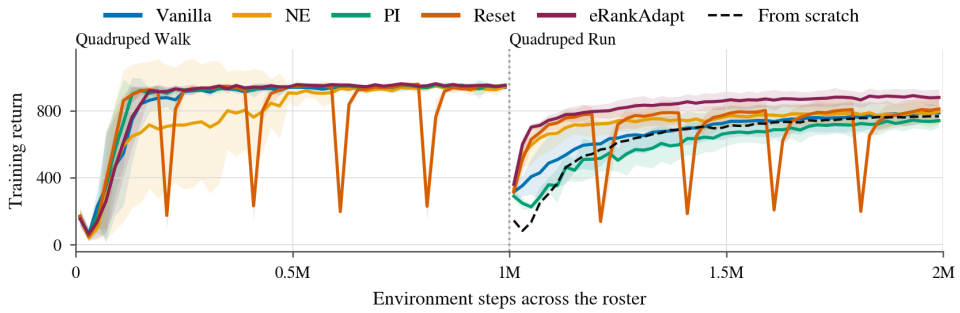


Figure 11: Sequential task episodic return in the `quadruped` environment. Results depict episodic return on the current task being trained. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

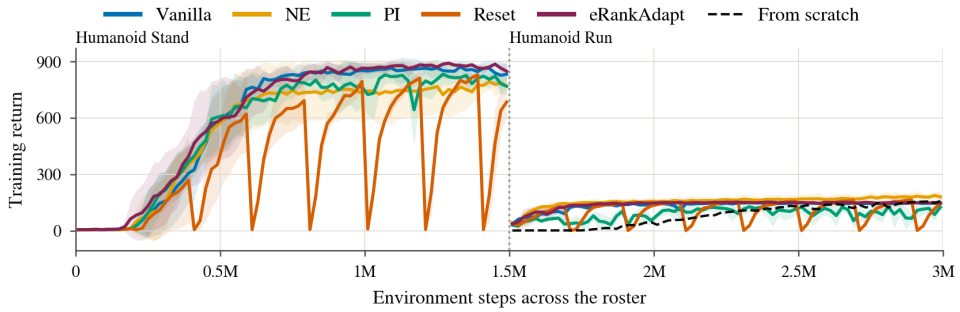


Figure 12: Sequential task episodic return in the `humanoid` environment. Results depict episodic return on the current task being trained. Solid lines and shaded areas show the mean and $1\times$ standard deviation, respectively.

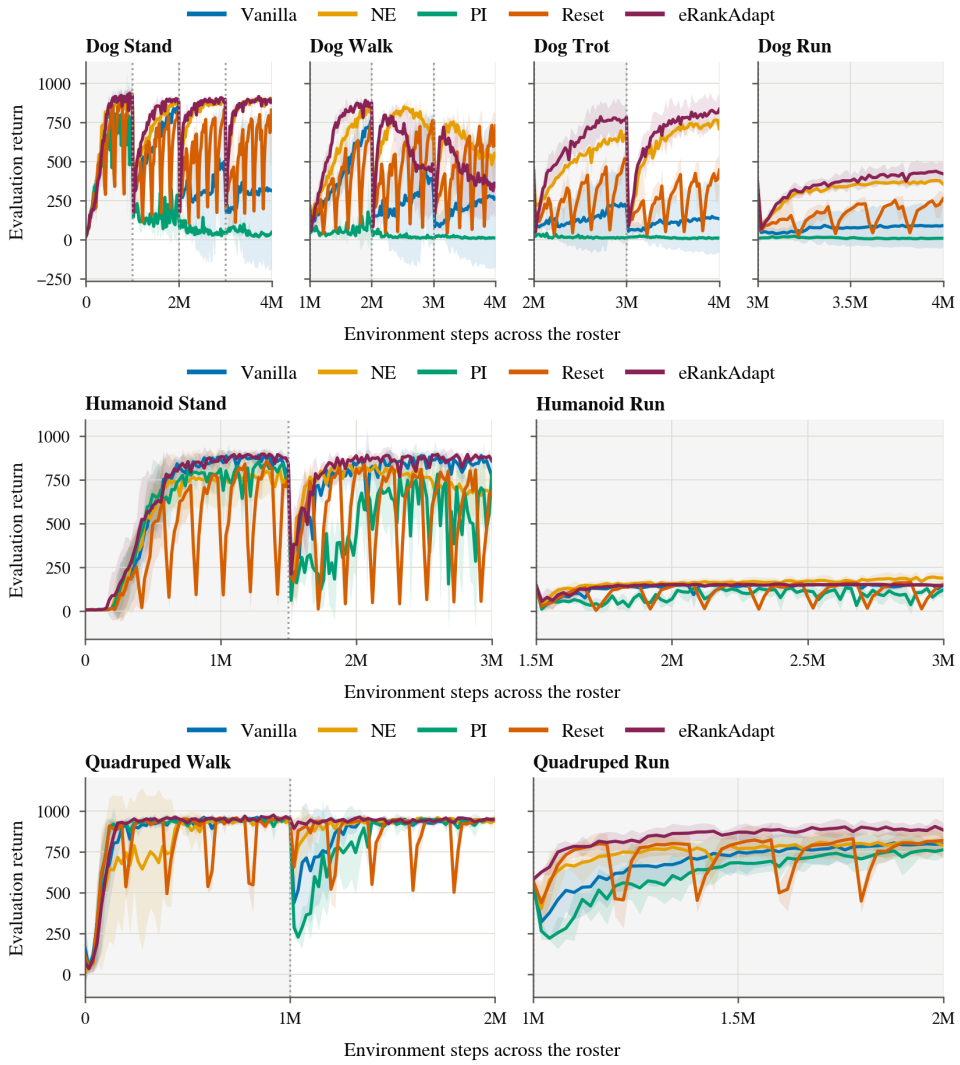


Figure 13: Evaluation episodic return on roster tasks, showcasing whether improving plasticity induces catastrophic forgetting on previously learned tasks.

pancy, we find that a smaller layer width yields higher occupancy, indicating it can leverage the width more effectively, whereas a larger layer width generally yields a higher effective rank.

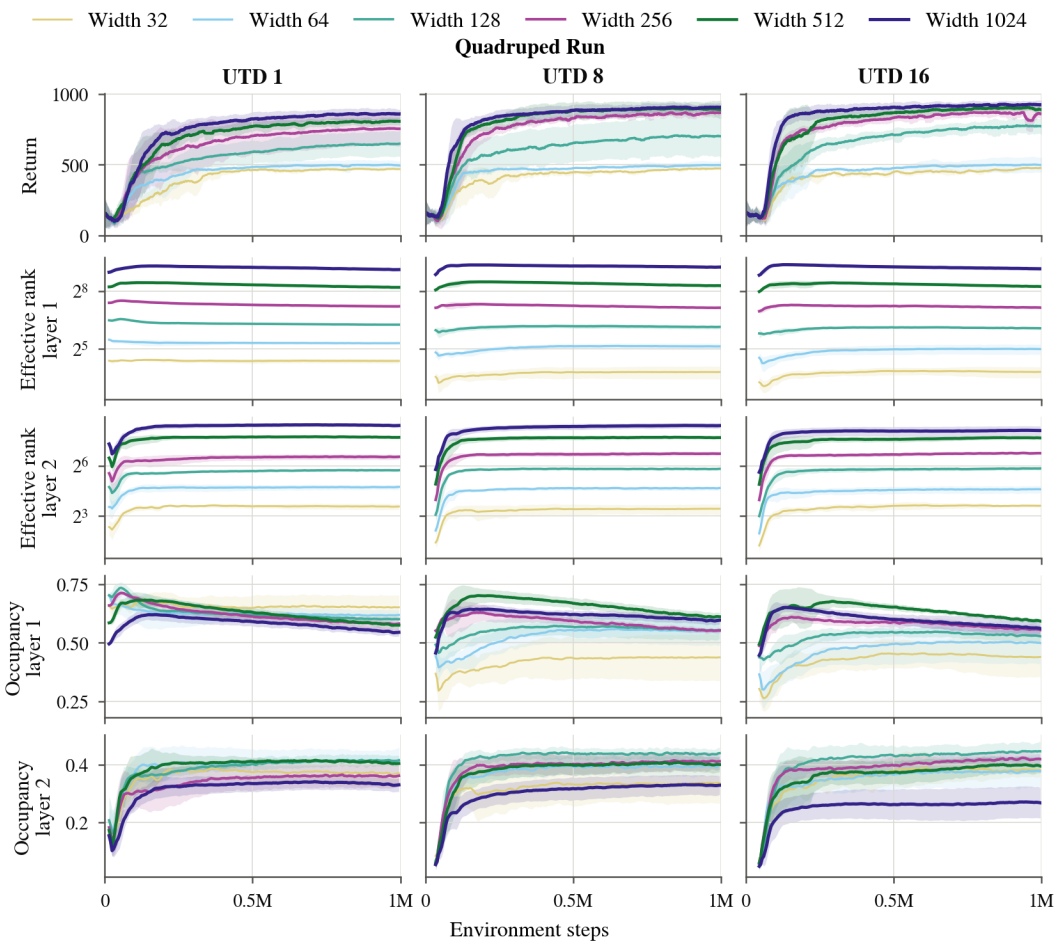


Figure 14: Performance and effective rank comparison of static width on environment `quadruped-run`. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

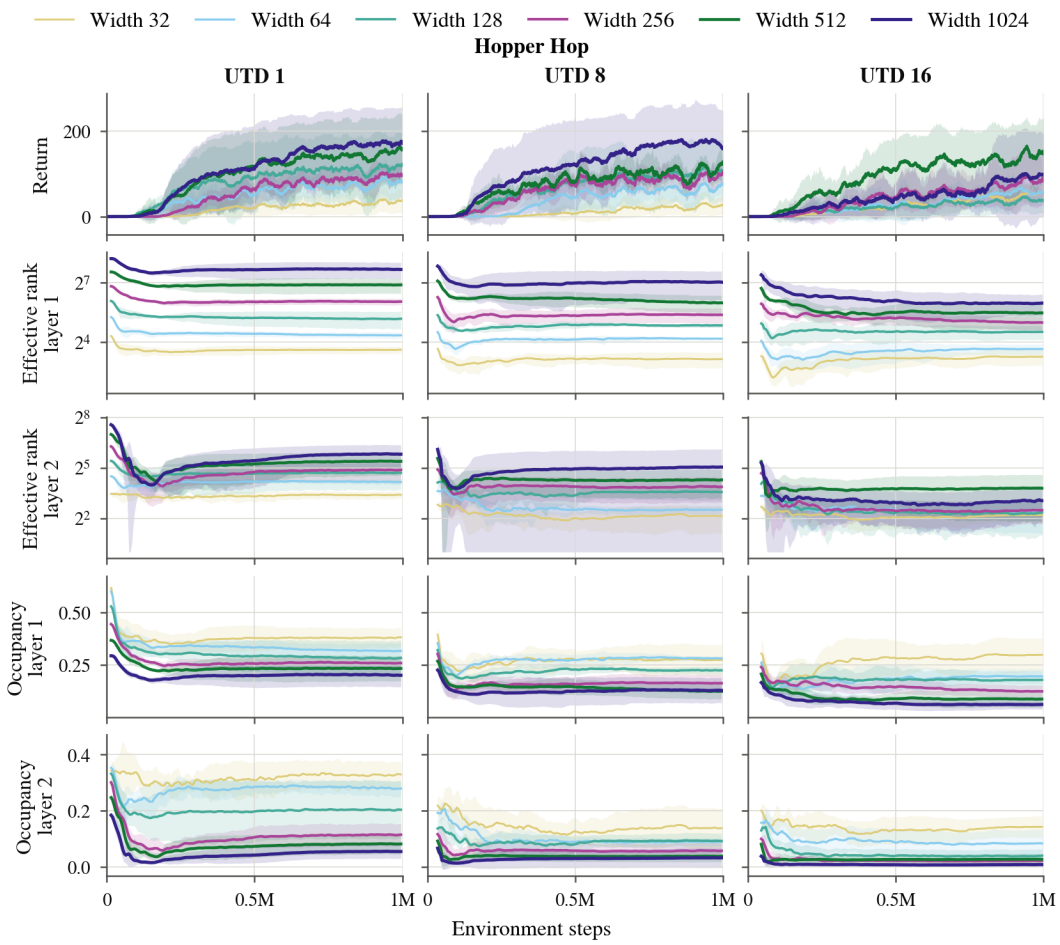


Figure 15: Performance and effective rank comparison of static width on `hopper-hop`. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

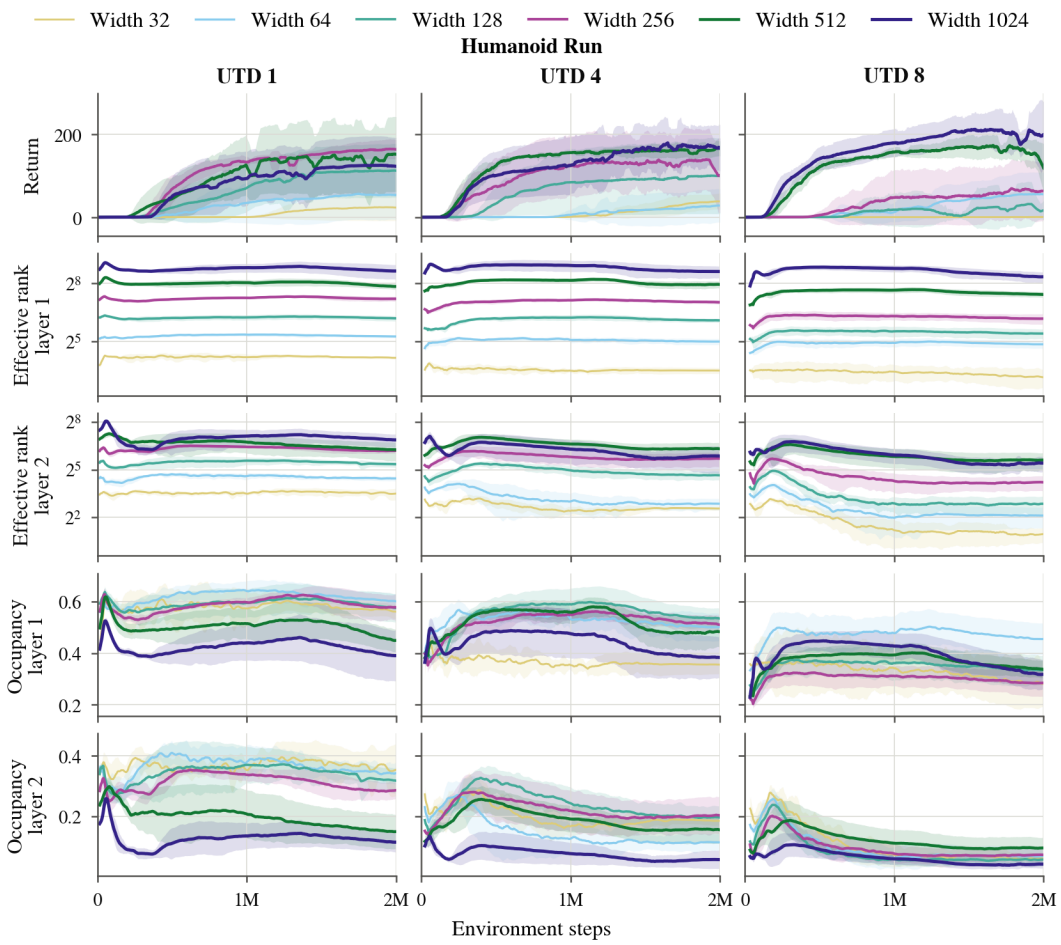


Figure 16: Performance and effective rank comparison of static width on `humanoid-run`. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

Table 1: Final hidden width of the growing critics of Layer 1 / 2 (neurons, mean ± std over seeds and SAC's two Q networks). UTD ratios per task in parentheses.

Task	Low UTD	Mid UTD	High UTD
`hopper` (1/8/16)	324±100 / 278±144	211±36 / 68±47	118±20 / 64±7
`quadruped` (1/8/16)	569±25 / 561±23	600±174 / 543±92	123±73 / 164±163
`humanoid` (1/4/8)	544±112 / 79±27	300±126 / 56±11	363±185 / 44±8

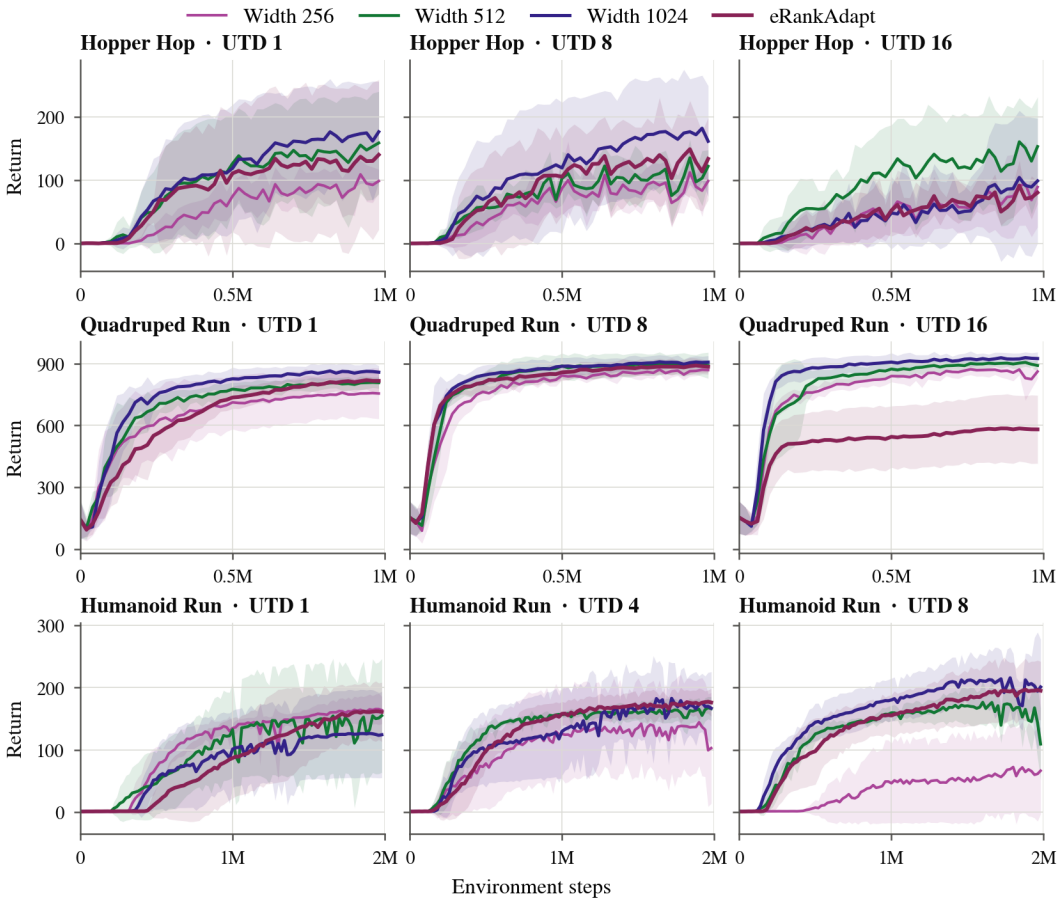


Figure 17: Performance comparison of static width with the proposed adaptive architecture (eRankAdapt). All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively. In most scenarios, the proposed adaptive architecture outperforms, or at least performs as well as, the best static network architecture.

F PRUNING THRESHOLD

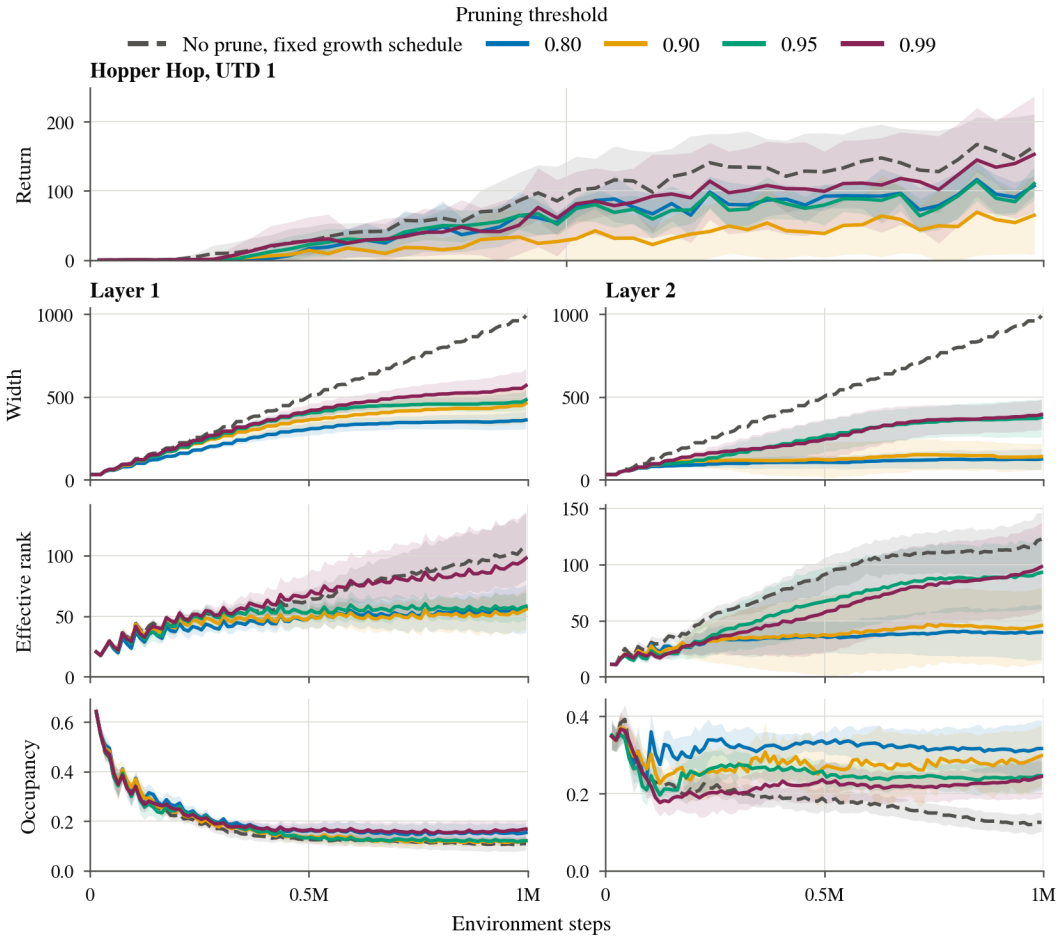


Figure 18: Performance and effective rank comparison of different pruning thresholds with a fixed growth schedule on environment `hopper-hop` at UTD ratio 1. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

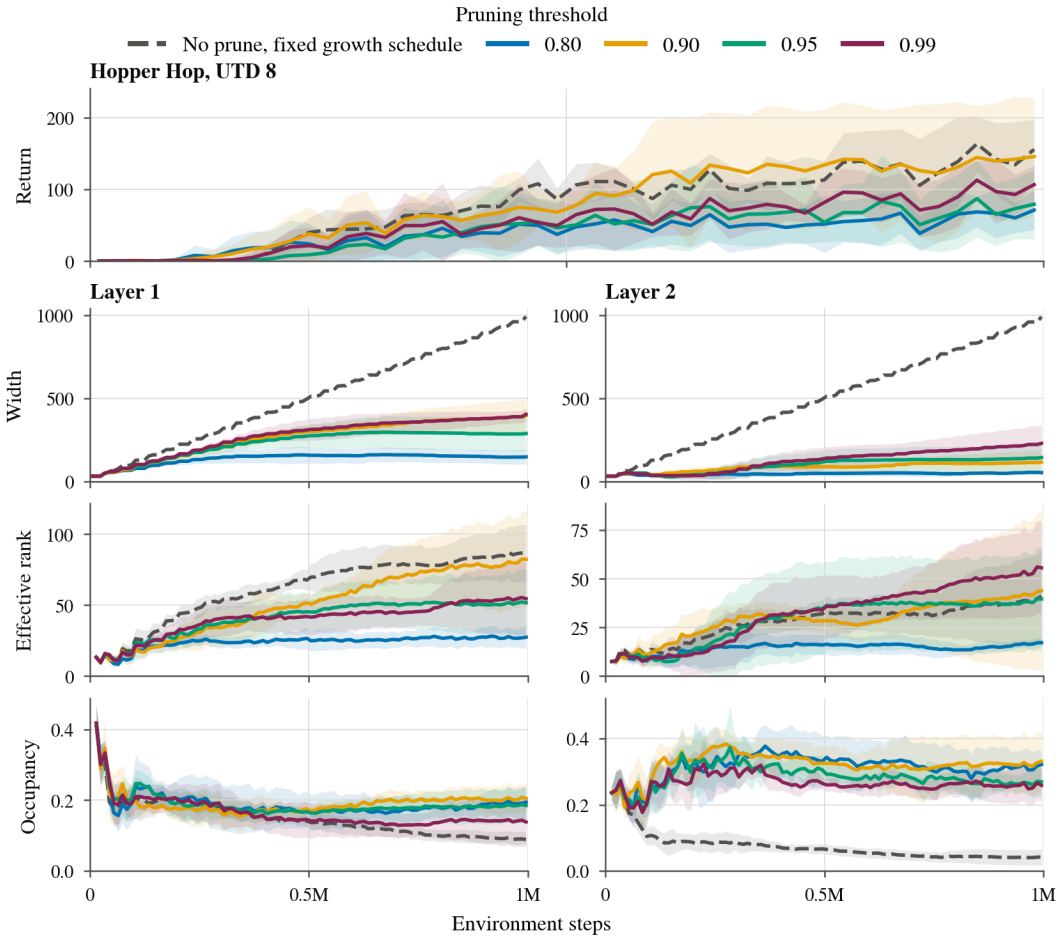


Figure 19: Performance and effective rank comparison of different pruning thresholds with a fixed growth schedule on environment `hopper-hop` at UTD ratio 8. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

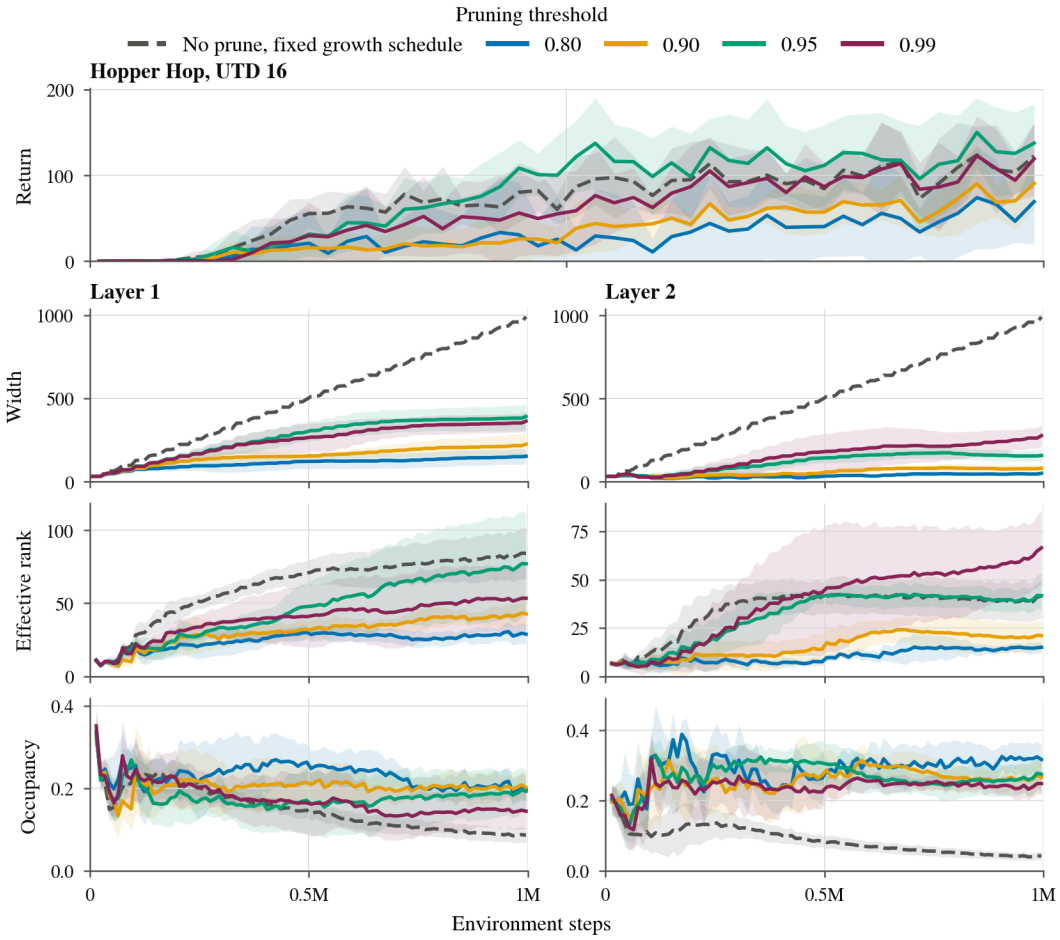


Figure 20: Performance and effective rank comparison of different pruning thresholds with a fixed growth schedule on environment `hopper-hop` at UTD ratio 16. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and 1× standard deviation, respectively.

G WEIGHT DECAY COEFFICIENT

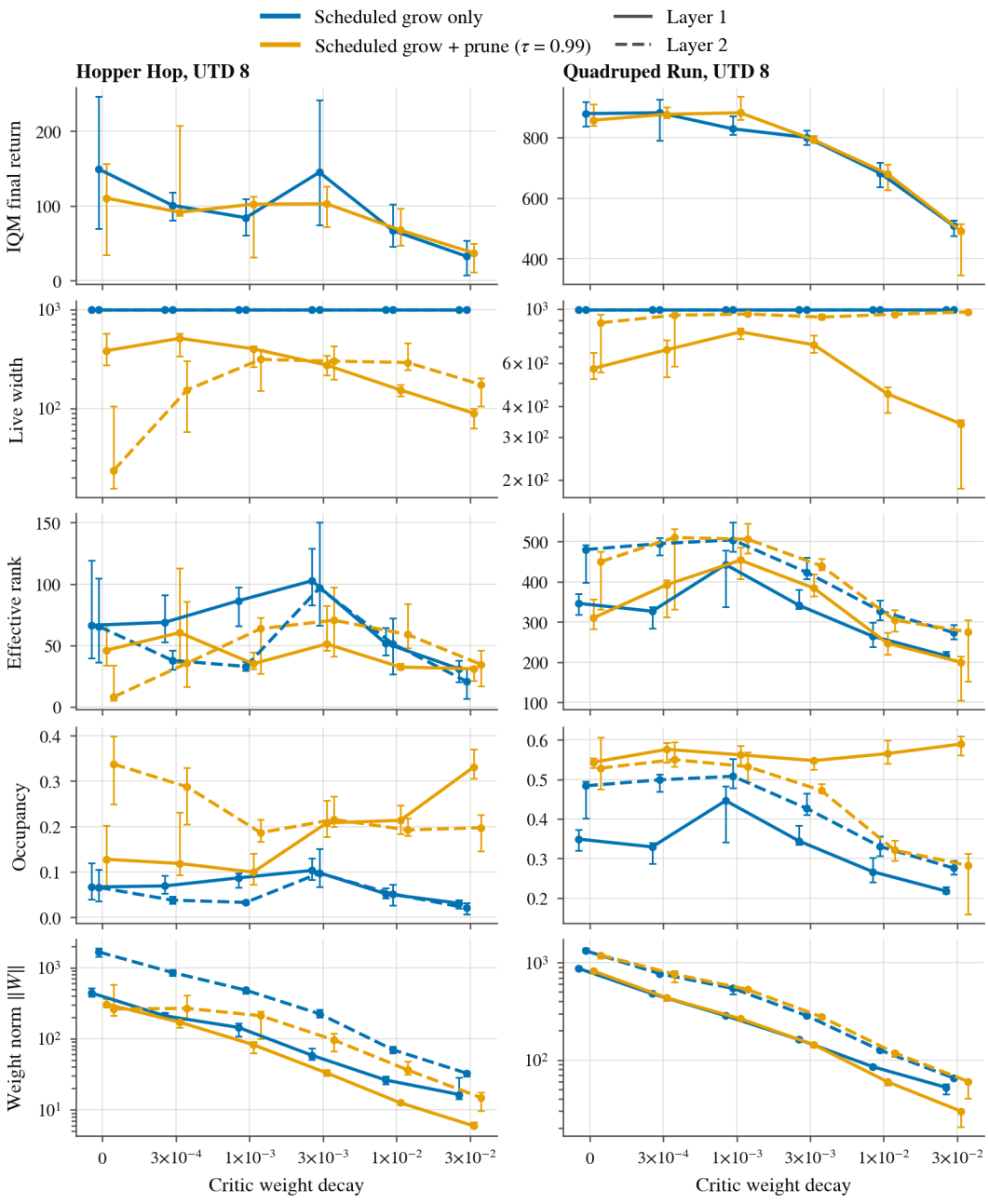


Figure 21: Performance and effective rank comparison with different weight decay coefficients with a fixed growth schedule and optional pruning with $\tau = 0.99$ on environments `hopper-hop` and `quadruped-run` at UTD ratio 8. We study $\lambda \in \{0, 3\times10^{-4}, 1\times10^{-3}, 3\times10^{-3}, 1\times10^{-2}, 3\times10^{-2}\}$. All experiments were conducted with five independent random seeds. Solid lines and shaded areas show the mean and $1\times$ standard deviation, respectively.