Addressing Sparse-Rewards in RL with Scalable Hierarchical Novel Eigen Options

Priyesh Vijayan, Élodie Côté-Gauthier, Mathieu Reymond, Sarath Chandar, Doina Precup, Isabeau Prémont-Schwarz

Keywords: sparse rewards, exploration, options, eigenoptions, Laplacian representations

Summary

Sparse-reward reinforcement learning often requires long, coherent exploration before any extrinsic reward is observed. We present Scalable Hierarchical Novel Eigen Options, a framework that scales novelty-directed spectral exploration to pixel-based domains by learning a novelty-weighted Laplacian representation online, converting learned eigen-potentials into option rewards, and training a hierarchical agent to deploy them as temporally extended exploratory actions.

A central challenge is that learned eigen-potential rewards can be directionally useful but locally unreliable under online function approximation, causing options to stall or loop. We address this by augmenting eigen-potential rewards with a novelty bonus, yielding longer and more reliable options that more consistently reach novel states. Empirically, we evaluate on Atari MONTEZUMA'S REVENGE, VENTURE, and seven hard ProcGen games, where our scalable variants outperform prior scalable Laplacian-option and direct NEO-style baselines.

Contribution(s)

1. We scale novelty-directed eigenoptions beyond tabular domains by learning a novelty-weighted Laplacian representation online from high-dimensional observations using a continuous graph-drawing objective with multistep and novelty-weighted training.
   Context: Prior NEO-style methods rely on exact tabular eigenvectors or explicit graph structure; this contribution concerns an approximate online representation-learning formulation for pixel-based reinforcement learning that prioritizes novel transitions and trajectories.
2. We introduce Hierarchical Novel Eigen Options (HNEO), which converts learned novelty-directed eigen-potentials into option rewards and trains a hierarchical Q-learning agent to select between primitive actions and temporally extended exploratory options.
   Context: The hierarchical control layer builds on standard options and SMDP value learning; the contribution is the integration of learned novelty-directed spectral option rewards into a scalable PQN-style hierarchical agent.
3. We identify local unreliability of learned eigen-potential rewards as a key obstacle to scaling novelty-directed spectral options, and propose RND-regularized eigen-options that augment eigen-potential rewards with novelty bonuses to reduce stalls and improve exploration.
   Context: The RND signal itself is not new; the contribution is using it as a reliability anchor for learned spectral option rewards, rather than only as a primitive-action intrinsic reward.
4. We provide empirical evidence that novelty-directed eigenoption methods can scale to hard pixel-based exploration benchmarks and, in the long run, outperform strong primitive-action novelty baselines such as RND.
   Context: To the best of our knowledge, this is the first eigenoption-based approach to show sustained improvements over RND-style primitive-action novelty exploration on the evaluated hard Atari and ProcGen benchmarks.

Addressing Sparse-Rewards in RL with Scalable Hierarchical Novel Eigen Options

Priyesh Vijayan[1,2], **Élodie Côté-Gauthier**[1,2], **Mathieu Reymond**[2], **Sarath Chandar**[2,3,5], **Doina Precup**[1,2,5], **Isabeau Prémont-Schwarz**[2,4]
priyesh.vijayan@mail.mcgill.ca

[1]**McGill University** [2]**Mila - Quebec AI Institute** [3]**Polytechnique Montreal** [4]**Université Laval** [5]**Canada CIFAR AI Chair**

Abstract

Temporally extended exploration via graph Laplacian-based options is a promising approach to sparse-reward reinforcement learning (RL), but existing methods either do not explicitly target novelty or fail to scale to pixel-based domains under function approximation. Novel Exploration via Orthogonality (NEO) addresses the first issue by constructing options that navigate from highly visited regions toward less visited ones, yet prior results were limited to settings where exact eigenvectors can be computed. We present a scalable extension of NEO to pixel-based domains, built on three contributions. First, we use a novelty-weighted continuous Laplacian graph-drawing objective, which enables RL with continuous observations. Second, we embed the resulting eigen-potential options within a hierarchical reinforcement learning framework, enabling coherent temporally extended behavior. Third, we observe that learned eigen-potential rewards are directional but locally unreliable under online approximation; we therefore augment each option reward with a novelty bonus, a novel design idea that proves essential for stabilizing option learning while preserving novelty-directed exploration. Together, these contributions yield stronger and more persistent exploration, enabling longer option rollouts and better access to hard-to-reach novel states. Empirically, our method significantly outperforms both the prior scalable Laplacian-option baseline and a direct extension of NEO on sparse-reward benchmarks under a fixed budget. On MONTEZUMA'S REVENGE, our best variant achieves approximately $1.8\times$ higher return than both baselines. On VENTURE, both baselines obtain returns near zero, while our method reaches a return of 1135. Across seven hard ProcGen games, our method achieves approximately $3.5\times$ and $5.6\times$ higher aggregate normalized return than the two baselines, respectively.

1 Introduction

Efficient exploration remains one of the central challenges in sparse-reward reinforcement learning (Jiang et al., 2023; Burda et al., 2018). In hard exploration environments, success often requires long, coherent action sequences before any useful extrinsic reward is observed. Novelty or visitation-count-based bonuses can help by rewarding the agent for reaching rarely visited states (Bellemare et al., 2016; Ostrovski et al., 2017; Burda et al., 2019), but their effect must still propagate through value-learning updates. This creates a lag between discovering novelty and acting purposefully to reach it. An alternative is to explore temporally extended actions, or *options* that can directly drive the agent toward informative regions of the state space (Sutton et al., 1999; Precup, 2000).

Graph Laplacian-based option discovery provides a principled framework for constructing temporally extended exploratory behaviors (Mahadevan & Maggioni, 2007; Machado et al., 2017). Such spectral methods use the eigenstructure of the state-transition graph to produce options that traverse the

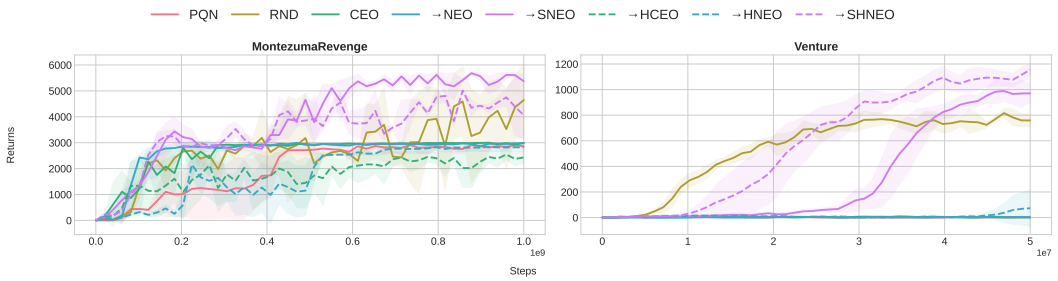


Figure 1: **Atari results.** Returns on MONTEZUMA'S REVENGE and VENTURE. Both scalable novelty-directed spectral variants, SNEO and SHNEO, outperform the prior scalable Laplacian-option baseline CEO. On MONTEZUMA'S REVENGE, the best scalable variant improves return from about 3000 to 5395 (1.8×). On VENTURE, CEO remains near zero, while the scalable variant reaches 1135.

environment along different topological directions and timescales (Machado et al., 2017; 2018; Jinnai et al., 2019). However, standard Laplacian options are primarily geometry-driven: they encourage coverage of the explored state-visitation graph but do not explicitly prioritize states that are currently underexplored. As a result, they may direct the agent toward distant but already familiar regions, rather than towards novel regions most useful for continued exploration.

Novel Exploration via Orthogonality (Theophilou & Şimşek, 2025), which we call Novel Eigen Options (NEO), addresses this limitation by modifying the Laplacian to incorporate visitation information directly into the spectral object. The resulting eigenvectors are biased to point away from highly visited regions, thereby converting novelty information into explicit multi-step exploratory guidance. Deep Covering Eigenoptions (CEO) (Klissarov & Machado, 2023) shows that Laplacian-based option discovery can be scaled beyond tabular domains by learning approximate eigenfunctions online, using graph-drawing objectives (Koren, 2003; Wu et al., 2019). However, the CEO also exposes a key difficulty of deep spectral options: eigenfunctions, option rewards, and option policies are all learned from a changing data distribution, so approximation noise can make spectral rewards locally unreliable. Straightforwardly scaling NEO amplifies this difficulty, since deep NEO must additionally learn a *novelty-weighted* Laplacian, in which the novelty estimates are themselves learned online and directly reshape the spectral object defining the option rewards. While the resulting eigen-potential rewards can remain directionally useful, they are locally noisy, causing option policies to drift, stall, or form local loops; in other words, these derived rewards are *unreliable*.

In this paper, we introduce Hierarchical Novel Eigen Options (HNEO), a framework for scaling novelty-directed spectral exploration to hard pixel-based environments. Like NEO, HNEO uses visitation information to bias the spectral structure toward underexplored regions of the state space. However, unlike NEO, which is restricted to tabular domains, it does not assume access to explicit neighbors or exact eigenvectors. Instead, HNEO learns a novelty-weighted Laplacian representation online (Koren, 2003; Gomez et al., 2024), uses the resulting eigen-potentials to define option rewards (Machado et al., 2017; Klissarov & Machado, 2023), and trains option policies inside a hierarchical agent that learns when to invoke them as temporally extended exploratory actions.

To address the derived reward reliability problem, we propose *Scalable (H)NEO*, which augments eigen-potential rewards with a novelty bonus (Burda et al., 2019). This helps options choose actions more likely to lead to genuinely novel states when the eigen-potential is ambiguous or corrupted, thereby reducing stalls and local loops while preserving spectral directionality. Therefore, while HNEO also uses novelty on the representation side, by weighting the Laplacian objective toward novel transitions and trajectories, Scalable (H)NEO stabilizes the option policies trained from the resulting eigen-potentials.

Contributions and overview: We present three main contributions:

- **Representation.** We scale NEO beyond tabular domains, by using a continuous graph-drawing objective based on ALLO (Gomez et al., 2024), which breaks the rotational symmetry between Laplacian eigenvectors. We extend this objective in two ways: by incorporating novelty weighting over transitions and trajectories, and by introducing a multi-step anchored smoothness term that exposes the representation to longer-range connectivity (Sec. 3).
- **Control.** We introduce *Hierarchical Novel Eigen Options (HNEO)*, which turns learned eigen-potentials into option rewards and trains a hierarchical agent to deploy these options as temporally extended exploratory actions (Sec. 4).
- **Reliability.** We identify unreliable eigen-potential rewards as a central obstacle to scaling novelty-directed eigen-options with online function approximation. To address this, we introduce *RND-regularized eigen-options*, which augment eigen-potential rewards with a novelty bonus (Burda et al., 2019) to stabilize option learning and improve exploration. We call this method *Scalable (H)NEO* (Sec. 5).

Together, these components yield a scalable framework for novelty-directed spectral exploration in hard, sparse-reward environments. Unlike prior scalable Laplacian-option methods, our approach avoids the early plateaus that limit deep spectral options and continues improving on difficult pixel-based Atari (Bellemare et al., 2013) and Procgen (Cobbe et al., 2020) tasks (Sec. 6). This provides, to the best of our knowledge, the first clear evidence that novelty-directed spectral options can scale to hard pixel domains; see Figures 1 and 3.

2 Preliminaries

We consider a discounted Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the primitive action space, $P(s'|s,a)$ is the transition kernel, $r(s,a,s')$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. A policy $\pi(a|s)$ induces trajectories $\tau = (s_0, a_0, r_0, s_1, \ldots)$.The corresponding action-value function is $Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t\geq 0}\gamma^t r_t | s_0 = s, a_0 = a]$. We consider methods that perform control by learning action values and acting (mostly) greedily with respect to them. We focus on sparse-reward and high-dimensional environments: observations are pixel-based, the transition graph is not given a priori, and both representation learning and control must be learned online.

Parallelized Q-learning. Our backbone learner is a hierarchical extension of *Parallelized Q-Networks* (PQN) (Gallici et al., 2025), a synchronous Q-learning method for vectorized environments. Unlike standard DQN-style methods, PQN does not rely on a large replay buffer or a separate target network; instead, it collects experience from I parallel environments over rollout windows of length T, forming a small online buffer from which it computes TD(λ) targets (Sutton & Barto, 2018) and performs batched updates. In the primitive action setting, the action-value network $Q_\theta(s,a)$ is trained toward TD(λ) returns; later, we will use the same backbone to learn values over temporally extended actions.

Options and SMDP value learning. We use the options framework (Sutton et al., 1999; Precup, 2000) to represent temporally extended exploratory behaviors. An option $o \in \mathcal{O}$ is a tuple (π_o, β_o) where $\pi_o(a|s)$ is the *intra-option* policy over primitive actions, and β_o is the termination function; in this paper, we consider that all options have a fixed duration n. For convenience, we treat primitive actions as length-one options. Consequently, we define $\mathcal{O}$ as an *ordered set*, in which the first $|\mathcal{A}|$ elements correspond to primitive actions. A high-level policy over options $\mu(o|s)$ selects a new option whenever the previous option terminates; the selected option then controls behavior by sampling $a_t \sim \pi_o(\cdot|s_t)$ at each primitive time step until termination according to β_o. If option o is initiated at s_t, runs for n steps, and accumulates discounted option reward $R_t^{(o)} = \sum_{j=0}^{n-1}\gamma^j r_{t+j}$, then SMDP Q-learning updates its value according to

$$Q(s_t, o) \leftarrow (1-\alpha)Q(s_t, o) + \alpha\Big(R_t^{(o)} + \gamma^n \max_{o'\in\mathcal{O}} Q(s_{t+n}, o')\Big). \tag{1}$$

In our hierarchical variant of PQN, high-level learning is performed over such option-induced transitions using a small online rollout buffer that contains state-option trajectories; we use TD(λ)-style updates where each high-level transition is considered one step for the purpose of computing the λ-return (which is natural for fixed-length options).

Behavior policy for option-based exploration. Following CEO (Klissarov & Machado, 2023), we collect data with an ϵ-μ-greedy behavior policy. With probability $1-\epsilon$, the agent acts greedily with respect to the current value function over options. With probability ϵ, it explores; in that case, it samples an extended option uniformly with probability μ, and otherwise samples a primitive action uniformly with probability $1-\mu$. This approach helps bias the agent towards sampling more extended options.

Source Laplacian and eigenoptions. Laplacian-based option discovery starts from a directed graph $G = (V, E)$ induced by experience, with random-walk transition matrix P. The standard random-walk Laplacian is $L_{\mathrm{rw}} = I - P$. Following NEO (Theophilou & Şimşek, 2025), we instead use a weighted *source Laplacian* $L_\zeta = I + \Gamma - P$, where Γ is diagonal with entries $\Gamma_{ss} = \zeta(s) \geq 0$. The weights are derived from visitation counts through the scaled count function $F(N(s)) = \delta(N(s)/\max_{s'} N(s'))^{1/k}$, where $\delta > 0$ sets the overall scale and $k > 0$ compresses the count range; we then set $\zeta(s) = F(N(s))$. In the tabular NEO construction, the smoothest eigenvectors of L_ζ induce gradients from highly visited states toward less visited ones. Given the m-th smoothest learned eigenfunction ϕ_m, we define its associated eigen-potential reward as $r_m(s_t, s_{t+1}) = \phi_m(s_{t+1}) - \phi_m(s_t)$. An eigenoption associated with ϕ_m is then trained to maximize this reward, that is, to ascend the corresponding eigen-potential and thus move toward states with a larger eigenfunction value.

RND-based novelty estimation. To estimate novelty and, in turn, visitation information from pixels, we use Random Network Distillation (RND) (Burda et al., 2019). Let f_{tgt} be a fixed randomly initialized target network and f_{prd} a trainable predictor; the prediction error $\mathrm{nov}(s) = \|f_{\mathrm{prd}}(s) - f_{\mathrm{tgt}}(s)\|_2^2$ serves as a novelty score, with larger values indicating less familiar states. In our method, this signal is used in three ways: first, to derive count-like quantities that parameterize the modified Laplacian; second, to weight the novel transitions and trajectories more heavily in the Laplacian eigenvector optimization; and third, to augment eigen-potential rewards when learning scalable option policies.

3 Scaling Novelty-Directed Options

This section develops the *representation layer* of our method. Our goal is to scale novelty-directed spectral structure from tabular NEO-style constructions to high-dimensional observation-based online RL, where the transition graph is not explicitly available and the eigenfunctions must be learned from sampled trajectories. We build on the graph-drawing view of Laplacian representation learning introduced by Wu et al. (2019), the ALLO (Gomez et al., 2024) objective for stable eigenfunction learning, and the suggestions by Theophilou & Şimşek (2025) in their appendix for extending the source Laplacian construction to symmetric continuous domains.

Vanilla graph drawing (Koren, 2003) Let $\phi : \mathcal{S} \to \mathbb{R}^d$ denote a parametric representation whose coordinates approximate the d smoothest eigenfunctions of a Laplacian operator. Although graph drawing is often introduced for finite graphs, its stochastic objective can be optimized directly from sampled transitions, thereby extending naturally to continuous, high-dimensional observations via a learned encoder ϕ. In the symmetric setting, the standard graph-drawing objective minimizes the Dirichlet energy subject to orthonormality constraints. In stochastic form, this can be written as

$$\mathcal{L}_{\mathrm{Lap}}(\phi) = \mathbb{E}_{s\sim\rho_\pi,\ s'\sim P_\pi(\cdot|s)} \left[\|\phi(s) - \phi(s')\|_2^2 \right]. \tag{2}$$

where ρ_π is the state distribution induced by the behaviour policy π and P_π is the one-step transition kernel under that policy. Eq. 2 encourages adjacent states to have similar embeddings, while the orthonormality constraints prevent collapse and separate different spectral directions.

Source-Laplacian extension. NEO (Theophilou & Şimşek, 2025) replaces the standard random-walk Laplacian with the source Laplacian $L_\zeta = I + \Gamma - P$, where the diagonal term Γ encodes novelty or visitation weighting. In the symmetric case, the corresponding Rayleigh quotient decomposes into the usual Laplacian smoothness term plus a weighted squared penalty on feature magnitude:

$$R(f) = \frac{\frac{1}{2}\sum_{i,j} P_{ij}\big(f_i - f_j\big)^2 + \sum_i \zeta_i f_i^2}{\sum_i f_i^2}. \tag{3}$$

This directly suggests a continuous-domain extension of vanilla graph drawing: keep the usual smoothness term and add the weighted penalty $\sum_i \zeta_i f_i^2$. Intuitively, states with larger ζ_i are penalized more heavily, so the smoothest eigenfunctions tend to place larger magnitude on states with smaller ζ_i, corresponding to less visited and more novel regions. The corresponding one-step NEO energy is therefore:

$$\mathcal{L}^{(1)}_{\text{NEO}}(\phi) = \mathbb{E}_{s\sim\rho_\pi,\, s'\sim P_\pi(\cdot|s)}\left[\|\phi(s) - \phi(s')\|_2^2\right] + \mathbb{E}_{s\sim\rho_\pi}\left[\zeta(s)\,\|\phi(s)\|_2^2\right]. \tag{4}$$

ALLO instantiation. To make the representation objective robust under function approximation, we combine the energy term with the same orthonormality-control mechanism as ALLO (Gomez et al., 2024). Let $\phi_k(s)$ denote the k-th coordinate of the learned representation, and let δ_{jk} be the Kronecker delta. Following ALLO, we use asymmetric stop-gradient constraints together with dual variables and a quadratic barrier. Writing $\text{sg}(\cdot)$ for the stop-gradient operator, the full objective is:

$$\mathcal{L}_{\text{rep}} = \mathcal{L}^{(1)}_{\text{NEO}} + \underbrace{\sum_{j=1}^{d}\sum_{k=1}^{j}\beta_{jk}\Big(\langle\phi_j, \text{sg}(\phi_k)\rangle - \delta_{jk}\Big) + b\sum_{j=1}^{d}\sum_{k=1}^{j}\Big(\langle\phi_j, \text{sg}(\phi_k)\rangle - \delta_{jk}\Big)^2}_{\mathcal{L}_{\text{orth}}}. \tag{5}$$

Here, the first term in $\mathcal{L}_{\text{orth}}$ is the linear dual penalty, with dual variables β_{jk}, and the second term is the quadratic barrier term with barrier coefficient b. The stop-gradient operator makes the constraints asymmetric, which is the key ALLO modification that breaks the rotational symmetry present in earlier graph-drawing objectives. In practice, the dual variables and barrier coefficient are updated online in a gradient-ascent fashion, following the max-min augmented-Lagrangian dynamics of ALLO, with updates driven by the linear and quadratic orthonormality violations.

Anchored multistep energy estimator. Our implementation does not directly optimize the one-step energy in Eq. (4). Instead, it optimizes a multistep source-Laplacian operator using sampled rollout segments. For each sampled rollout row i, we anchor the loss at a reference state $s_i^{(0)}$ and compare it to future states $s_i^{(1)}, \ldots, s_i^{(n)}$ from the same segment, where n is bounded by the rollout horizon. Although this notation may suggest a fresh trajectory start, in practice, these states come from an online PQN rollout buffer, and $s_i^{(0)}$ should be understood simply as the anchor state of the sampled segment. We use only the starting state of each rollout row as the anchor, rather than averaging the same multistep objective over all possible anchors within the segment, since the denser variant would induce many more Laplacian updates per PQN epoch and, in practice, cause the representation to evolve too quickly relative to option learning and Q-learning.

Let $\alpha_{i,k}$ denote normalized within-segment smoothness weights over future offsets, and let w_i denote normalized row weights across the batch. Then the energy term used by our method is:

$$\mathcal{L}^{(n)}_{\text{NEO}} = \frac{1}{B}\sum_{i=1}^{B} w_i \left[\sum_{k=1}^{n}\alpha_{i,k}\left\|\phi\Big(s_i^{(0)}\Big) - \phi\Big(s_i^{(k)}\Big)\right\|_2^2 + \frac{1}{n+1}\sum_{k=0}^{n}\zeta\Big(s_i^{(k)}\Big)\left\|\phi\Big(s_i^{(k)}\Big)\right\|_2^2\right], \tag{6}$$

where the first term is a multistep anchored smoothness loss, while the latter defines the source-Laplacian sink penalty over valid future states.

This multistep construction is motivated by the fact that the corresponding multistep operator can be written as a polynomial in the underlying one-step operator and therefore shares the same eigenvectors

while transforming only the eigenvalues. We show in Appendix B that this yields a principled extension of the one-step objective. Empirically, we observe that multistep training improves cosine similarity to the ground-truth eigenvectors more rapidly in the early stages of learning.

Within-segment and row weighting. The weights $\alpha_{i,k}$ determine which future offsets contribute most to the multistep smoothness term. In our method, we combine temporal discounting with future-state novelty:

$$\alpha_{i,k} = \frac{\gamma^{k-1}\,\mathrm{nov}(s_i^{(k)})}{\sum_{\ell=1}^{n}\gamma^{\ell-1}\,\mathrm{nov}(s_i^{(\ell)})}, \qquad k = 1,\dots,n, \tag{7}$$

where $\mathrm{nov}(s)$ denotes the novelty score assigned to state s obtained from RND. Thus, future states that are both temporally near and highly novel receive a larger contribution within each sampled rollout segment.

At the sample level, we further emphasize rollout segments that contain highly novel future states. For each sample/rollout i, we define its novelty mass by the maximum novelty among its valid future states and normalize these values across the batch to obtain row weights.

$$w_i = \frac{B\,m_i}{\sum_{j=1}^{B} m_j}, \qquad m_i = \max_{1\le k\le n} \mathbf{1}_{i,k}\,\mathrm{nov}(s_i^{(k)}) \tag{8}$$

Hence, $\alpha_{i,k}$ controls which future offsets matter within a rollout segment, while w_i controls which sampled segments receive larger weight across the batch.

Calibrating novelty into source-Laplacian weights. The source term in Eq. (6) requires nonnegative weights $\zeta(s)$ that play the role of visitation-derived source coefficients in NEO. In our implementation, these are not obtained from explicit visitation counts, which are unavailable in high-dimensional pixel domains, but from RND-based inverse counts. Concretely, let $\mathrm{nov}(s)$ denote the normalized RND novelty bonus and define the inverse-count proxy $\mathrm{inv}(s) = 1/(\mathrm{nov}(s)+\varepsilon)$. We then normalize this proxy by a running maximum, map it through the NEO scaling rule, $\zeta(s) = \delta\left(\frac{\mathrm{inv}(s)}{\max \mathrm{inv}}\right)^{1/k}$. We additionally use an online mean calibration so that the average source weight remains close to a fixed target value throughout training. This stabilizes the overall strength of the source penalty even as the novelty distribution changes; see appendix D for exact details.

Why ALLO remains useful in directed environments Although graph-drawing objectives are often viewed as tools for undirected graphs, the construction of Wu et al. (2019) implicitly makes the directed transition kernel reversible, by combining it with its stationary time-reversal. As shown in Appendix C, the resulting operator is similar to the normalized directed Laplacian in (Chung, 2005), making ALLO-style learning applicable to directed environments such as Atari and ProcGen. Moreover, our results show empirically that these methods, though originally designed for static graphs, remain useful in practice even for non-stationary policy-induced graphs.

4 Hierarchical Eigen Options

We now describe the *control layer* of our method, which turns the learned novelty-directed spectral representation into temporally extended exploratory behavior inside a hierarchical PQN agent. The key idea is simple: the Laplacian network produces eigenfunctions, the option policies are trained to follow the corresponding eigen-potential rewards introduced in Section 2, and a high-level controller learns when to invoke these options versus when to act with primitive actions directly. These component together define Hierarchical Novel Eigen Options (HNEO).

Hierarchical PQN with a persistent high-level buffer. At the primitive time scale, the agent interacts with I parallel environments for rollout windows of length T, exactly as in PQN. Primitive transitions are used to update the primitive-action Q-values, the Laplacian representation, and the option policies. In parallel, we maintain a *persistent* buffer of length M that stores completed high-level transitions of the form $(s_t, o_t, R_t, l_t, s_{t+l_t})$, where t denotes the primitive-time index at

which the option starts, l_t is the duration of the option, and $R_t = \sum_{j=0}^{l_t-1} \gamma^j r_{t+j}$ is the option-level reward accumulated during execution. This buffer must persist across primitive rollouts, since an option may begin in one PQN rollout window and terminate only in a later one. Thus, while the primitive rollout data is refreshed every update, this is a rolling SMDP buffer over completed options.

SMDP TD(λ) We train the high-level controller with an SMDP TD(λ) target computed from the persistent SMDP buffer. For a transition $(s_t, o_t, R_t, l_t, s_{t+l_t})$, let $\Gamma_t = \gamma^{l_t}$ denote the segment discount and let $V(s) = \max_o Q(s, o)$. The meta-target is then computed by the backward recursion

$$G_t = R_t + \Gamma_t \Big((1-\lambda) V(s_{t+l_t}) + \lambda G_{t+1} \Big), \tag{9}$$

with $G_t = R_t$ on terminal transitions. This is the standard TD(λ) recursion applied to SMDP transitions, with one practical simplification: while the discount factor is duration-aware through $\Gamma_t = \gamma^{l_t}$, the trace ignores the option duration l_t and considers each SMDP transition to be single step. This is possible because all options have fixed duration from all states.

Primitive and option-policy updates. The rest of the model is trained from the primitive rollout buffer in the usual online PQN style: primitive action values are updated with standard TD(λ), the Laplacian eigenfunctions are updated with the multi-step novelty-weighted representation loss from Sec 3, and the intra-option policies are updated from primitive rollout data using their eigen-potential rewards.

5 RND-regularized Eigen-Options

(H)NEO trains options using eigen-potential rewards. In the deep online setting, these rewards can be unreliable because both ingredients that shape them are non-stationary: the Laplacian representation is learned online from a small changing buffer, and the source-Laplacian novelty weights are produced by an RND model that is also updated online. As a result, the local option reward

$$r_m(s_t, s_{t+1}) = \phi_m(s_{t+1}) - \phi_m(s_t)$$

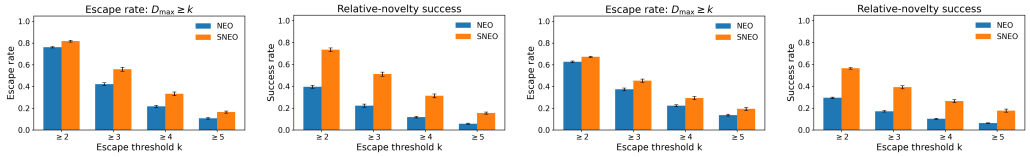
may contain spurious local extrema even when the learned eigenfunction captures a useful large-scale exploratory direction, causing options to stall, drift, or enter short local loops.

RND-regularized option reward: To stabilize option learning, we augment the eigen-potential reward with a normalized RND novelty score:

$$r_m^{\text{rel}}(s_t, s_{t+1}) = \big(\phi_m(s_{t+1}) - \phi_m(s_t)\big) + \beta_{\text{rnd}}\, \text{nov}(s_{t+1}), \tag{10}$$

The two terms play complementary roles. The eigen-potential is dense and directional, providing local gradients aligned with a learned large-scale path toward novelty, but under online function approximation, it can be locally noisy. The RND bonus is less directional but more reliable as a novelty signal, helping the option prefer local directions that are more likely to lead to genuinely novel states when the eigen-potential alone is ambiguous or corrupted. Thus, the RND term acts as a reliability anchor for option learning, reducing stalls and local loops while preserving the exploratory direction encoded by the eigenfunction. We refer to option policies trained with Eq. (10) as *RND-regularized eigen-options*. Since these RND-regularized eigen-options are the key ingredient that makes novelty-directed eigen-options scalable in pixel-based domains, we refer to the resulting non-hierarchical method as *Scalable NEO* (*SNEO*) and its hierarchical variant as *Scalable HNEO* (*SHNEO*). See appendix E for the corresponding algorithms.

Mechanistic analysis of RND-regularized eigen-options To illustrate the effect of using RND-regularized eigen-options, we use two gridworlds, GridRoom-16 (GR16) and GridMaze-19 (GM19). Because these custom environments are not parallelized for PQN, we use the same representation-learning components inside a DQN training loop for this diagnostic. At $10{,}000$ environment steps, we extract the learned eigenfunctions and RND novelty map. For every start state and option dimension, we compare greedy rollouts of length 25 under the vanilla novel-eigen-options (NEO)



(a) GR16: escape (b) GR16: relative novelty (c) GM19: escape (d) GM19: relative novelty

Figure 2: Mechanistic comparison of Scalable NEO with and without RND-regularized eigen-options. Using RND-regularized eigen-options consistently improves escape, and the gains are larger for relative-novelty success.

and RND-regularized eigen-options (SNEO). The models have 5 options, each with a duration of 5 time steps.

We report two diagnostics: (i) *Escape*, which measures whether a rollout reaches maximum Manhattan (ℓ_1) distance (Deza & Deza, 2009) at least k from its start state, $\mathrm{Escape}_{\geq k} = \mathbf{1}\{D_{\max} \geq k\}$; and (ii) *Relative-novelty success*, which additionally requires the final state to be more novel than the start state, $\mathrm{RelSucc}_{\geq k} = \mathbf{1}\{D_{\max} \geq k\}\, \mathbf{1}\{\mathrm{nov}(s_{\mathrm{final}}) > \mathrm{nov}(s_0)\}$.

As shown in Fig. 2, RND-regularized eigen-options improve both diagnostics in both environments. On GR16, the average escape rate across $k \in \{2, 3, 4, 5\}$ increases from 0.37 to 0.45, a 21.6% relative gain. Relative-novelty success increases from 0.19 to 0.40, a 110% gain. On GR19, escape increases from 0.35 to 0.40, a 14.7% gain, while relative-novelty success increases from 0.17 to 0.35, a 111% gain. The larger improvement in relative-novelty success suggests that the RND-regularized option reward does not merely extend option trajectories; it improves local decision-making so that the resulting climb is more likely to terminate in states that are genuinely novel relative to where the option began. We provide additional analysis of the endpoint distance between the learned options and the exact source-Laplacian option endpoints in Appendix F.

6 Experiments

We evaluate whether our method improves sample-efficient exploration in hard sparse-reward settings in both ProcGen (Cobbe et al., 2020) and Atari (Bellemare et al., 2013) environments. Since our focus is exploration rather than generalization, we evaluate on ProcGen in the *same-level* setting under a hard distribution, following the recommendation to match train and test environments when studying exploration. We first ran vanilla PQN (Gallici et al., 2025) on ProcGen and selected environments in which it failed to achieve a normalized score of 1 within the training budget. This yielded a subset of seven hard environments: BOSSFIGHT, CLIMBER, FRUITBOT, HEIST, JUMPER, LEAPER, and NINJA. We train all ProcGen agents for 25M environment steps and report normalized returns.

For Atari, we applied the same selection procedure as for ProcGen: we ran vanilla PQN across the sparse-reward Atari suite and retained environments in which it failed to achieve a nonzero return within the training budget. This yielded three hard games: VENTURE, PITFALL!, and PRIVATE EYE. However, we exclude PITFALL! and PRIVATE EYE as uninformative: in PITFALL!, all methods, including strong count- and novelty-driven baselines, obtain zero return at best throughout training (Badia et al., 2020b), so no meaningful comparison between methods is possible. In PRIVATE EYE, all methods perform equally poorly, providing no discriminative signal with which to evaluate exploration strategies (Berseth, 2025). Thus, we report results only on VENTURE and additionally evaluate on MONTEZUMA'S REVENGE, the canonical hard-exploration benchmark, using a budget of 1B environment steps. VENTURE is trained for 50M environment steps.

All methods use PQN as the backbone RL learner. Our baselines are vanilla PQN (Gallici et al., 2025), PQN with RND intrinsic bonus (RND) (Burda et al., 2019), Deep Covering Eigen Options (CEO) (Klissarov & Machado, 2023), and Novel Eigen Options naively scaled up with a continuous graph-drawing objective (NEO) (Theophilou & Şimşek, 2025). We also consider hierarchical versions of CEO and NEO, denoted HCEO and HNEO. We compare these against our proposed scalable

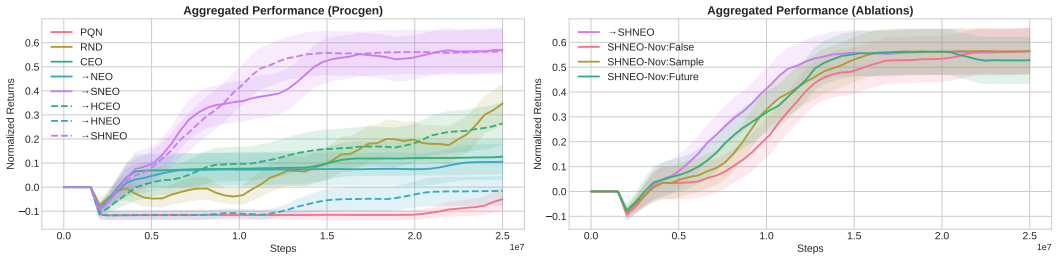


Figure 3: **Aggregate ProcGen results.** Left: aggregate normalized returns on ProcGen Hard-7. Right: ablation of novelty weighting within the multi-step representation loss.

variants, SNEO and SHNEO. For all option-based methods, we use a shared convolutional encoder and separate feed-forward Q-networks per option. Unless otherwise stated, we use 10 options in all environments and fix the option duration to 5 time steps in ProcGen and 10 time steps in Atari. Results are averaged over 5 random seeds, except for the 1B-step Montezuma runs, which are averaged over 3 seeds. Full implementation details are provided in Appendix H. Note that the full code will be released upon acceptance of this manuscript.

Main results on Procgen Hard-7: Figure 4 shows the per-environment normalized return curves on the Procgen Hard-7 benchmark, and Figure 3 (left) shows the aggregate normalized return across the same set. The aggregate result is clear: the two scalable novelty-directed spectral methods, SHNEO and SNEO, are the strongest overall, outperforming the next best baseline, RND, by a wide margin. Between these two, however, the advantage is environment-dependent, with SHNEO only slightly ahead in the aggregate.

Looking first at the games, the seven environments separate naturally into two groups. Agents reach a normalized score of 1 only on four games: CLIMBER, HEIST, JUMPER, and NINJA. The remaining three games, BOSSFIGHT, FRUITBOT, and LEAPER, are not solved by any method within the 25M-step budget. In BOSSFIGHT and FRUITBOT there is partial progress, with several methods consistently obtaining non-zero reward, whereas in LEAPER, all methods remain at zero throughout training. Thus, even within the Hard-7 benchmark, there is a clear distinction between environments where exploration eventually succeeds and environments that remain beyond reach for the entire family of methods we examined.

Looking next at the methods, the scalable variants SNEO and SHNEO are the only methods that consistently solve all four solvable games. They are also the two most sample-efficient methods overall, with especially large margins on JUMPER and NINJA, followed by CLIMBER. On HEIST, the ranking is more nuanced: several methods besides PQN and HNEO eventually reach a normalized score of 1, and while the scalable variants are not the top two by sample efficiency there, they still solve the task early and remain among the strongest performers.

HEIST and FRUITBOT are the only environments in which CEO and NEO are among the most sample-efficient methods. Outside these cases, however, NEO and its hierarchical version HNEO perform poorly, often remaining near zero throughout training. In contrast, CEO and HCEO are substantially stronger and more stable than NEO and HNEO, and HCEO generally improves over CEO. This comparison is important: it shows that novelty-directed spectral rewards are not automatically beneficial when learned online from pixels. Without the RND-regularized eigen-options, NEO-style option rewards can be brittle enough to underperform even standard covering eigen-options.

Finally, RND is the third strongest method overall. It performs consistently on the four solvable games, but at a much slower rate than SNEO and SHNEO; thus, without the directional guidance provided by the learned eigen-potentials, it is markedly less sample-efficient. The gap between RND and the reliable spectral methods we proposed supports our central claim: reliable novelty estimation is useful, but combining it with novelty-directed spectral guidance yields substantially better results.

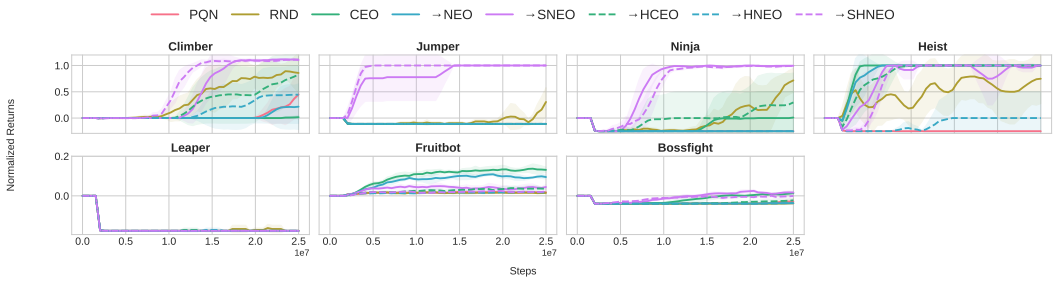


Figure 4: **Procgen Hard-7 results.** Normalized returns on the seven selected Procgen environments.

Ablation: novelty weighted Source Laplacian Learning Figure 3 (right) isolates the effect of the *novelty weighting inside the multi-step Laplacian loss*, rather than the source-Laplacian ζ weights themselves. We compare four variants: no novelty weighting, future-state novelty weighting only, sample-level novelty weighting only, and the full combination. The full method performs best overall, indicating that the two weighting mechanisms are complementary. Future-state weighting determines which offsets within a rollout segment contribute most to the smoothness term, whereas sample-level weighting emphasizes segments that contain highly novel states. Together, these results support the design of the multi-step novelty-weighted representation loss in Sec. 3.

We also report sensitivity to the number of options in Appendix G.

Main results on Atari: Figure 1 (left) shows results on MONTEZUMA'S REVENGE. This is the clearest Atari test of extreme long-horizon sparse exploration, and it exhibits the same overall pattern seen in ProcGen: the reliable novelty-directed spectral methods are strongest overall. In particular, SNEO and SHNEO perform similarly and clearly outperform the remaining baselines. Pure RND also learns eventually but is substantially noisier and less sample-efficient. In contrast, CEO, NEO, and vanilla PQN plateau substantially lower. Thus, in MONTEZUMA'S REVENGE, directional spectral guidance is clearly useful, but it becomes much more effective when stabilized by the RND-regularized option reward.

Fig. 1 (right) shows results on VENTURE. Here, SHNEO is the strongest method by the end of training, while SNEO also performs well and clearly improves over the non-reliable spectral baselines. RND makes progress earlier than most methods and remains competitive for much of the training, but is rapidly overtaken by SNEO and SHNEO. In contrast, CEO, NEO, HNEO, and PQN remain weak throughout. This suggests that novelty alone can drive some progress in VENTURE, but reliable spectral guidance yields the strongest long-horizon exploration.

7 Discussion, Limitation, and Conclusion

Our results show that novelty-directed spectral options can scale beyond tabular domains when paired with two ingredients: a scalable online representation objective and a reliable option reward. More broadly, we recommend Scalable (H)NEO as a wrapper around global count-based or lifelong novelty methods, where the underlying novelty estimator identifies persistent, underexplored regions and the spectral option layer turns that signal into directed, temporally extended exploratory behavior. The main limitation of this formulation is that it is naturally matched to global novelty signals and therefore does not extend as directly to episodic bonuses, whose reset-dependent within-episode nature does not define a stable global source-Laplacian structure.

Overall, we view scalable novelty-directed spectral option learning as a promising interface layer for global exploration signals. Hybrid intrinsic methods that include exploitation-oriented signals, such as contribution-style bonuses (Yang et al., 2024), appear especially compatible with it and could be integrated directly into the reliable option reward. We hope that this work encourages further combinations with stronger global novelty estimators, hybrid intrinsic objectives, and more efficient forms of hierarchical control.

References

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pp. 507–517. PMLR, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, et al. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2020b.

Amitay Bar, Ronen Talmon, and Ron Meir. Option discovery in the absence of rewards with manifold analysis. In *International Conference on Machine Learning*, pp. 664–674. PMLR, 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.

Glen Berseth. Is exploration or optimization the problem for deep reinforcement learning? *arXiv preprint arXiv:2508.01329*, 2025.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9 (1):1–19, 2005.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Michel Marie Deza and Elena Deza. Encyclopedia of distances. In *Encyclopedia of distances*, pp. 1–583. Springer, 2009.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *The Eleventh International Conference on Learning Representations*, 2023.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Diego Gomez, Michael Bowling, and Marlos C Machado. Proper laplacian representation learning. *International Conference on Learning Representations(ICLR)*, 2024.

Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via elliptical episodic bonuses. *Advances in Neural Information Processing Systems*, 35:37631–37646, 2022.

Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking exploration. *Royal Society Open Science*, 10(6):230539, 2023.

Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *International Conference on Machine Learning*, pp. 3130–3139. PMLR, 2019.

Martin Klissarov and Marlos C. Machado. Deep Laplacian-based options for temporally-extended exploration. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17198–17217. PMLR, 23–29 Jul 2023.

Yehuda Koren. On spectral graph drawing. In *International Computing and Combinatorics Conference*, pp. 496–508. Springer, 2003.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning. In *International conference on machine learning*, pp. 2295–2304. PMLR, 2017.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Bk8ZcAxR-.

Marlos C Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of machine learning research*, 24(80):1–69, 2023.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10), 2007.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor options: an option discovery framework for reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3304–3310, 2019.

Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphael Marinier, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The mit press edition, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Andreas Theophilou and Özgür Şimşek. Novel exploration via orthogonality. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Priyesh Vijayan, Padideh Nouri, Rishav Rishav, Sarath Chadar, Yash Chandak, Mathieu Reymond, Samira Kahou, and Doina Precup. Revisiting laplacian representations for value function approximation in deep rl. In *Inductive Biases in Reinforcement Learning Workshop at RLC 2025*, 2025.

Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pp. 11003–11012. PMLR, 2021.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJlNpoA5YQ.

Kai Yang, Jian Tao, Jiafei Lyu, and Xiu Li. Exploration and anti-exploration with distributional random network distillation. In *International Conference on Machine Learning*, pp. 56397–56421. PMLR, 2024.

Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Wenjun Zeng, and Glen Berseth. Rlexplore: Accelerating research in intrinsically-motivated reinforcement learning. *arXiv preprint arXiv:2405.19548*, 2024.

A Related Work

Spectral and dynamics-based representations. A long line of work studies state representations derived from operators induced by the environment dynamics. Proto-value functions (PVFs) introduced the use of Laplacian eigenvectors as basis functions for representation learning and control in MDPs, grounding representation quality in the geometry of the transition graph (Mahadevan & Maggioni, 2007). More recent work extended this idea to deep RL. Proto-value networks (PVN) scale the proto-value perspective through auxiliary prediction tasks based on the successor measure, showing that dynamics-aware spectral-style features can remain useful under function approximation (Farebrother et al., 2023). In parallel, Wu et al. proposed a scalable graph-drawing objective for learning Laplacian eigenfunction approximations directly from sampled transitions, and later work improved the stability and identifiability of such objectives (Wang et al., 2021; Gomez et al., 2024). A later work used these Laplacian embeddings as state representations and state-action representations for PQN (Vijayan et al., 2025).

Spectral option discovery. Spectral option discovery methods build exploratory behaviors from structure induced by the environment dynamics. Eigenoptions use Laplacian eigenvectors to define intrinsic rewards whose policies move along major directions of the state graph (Machado et al., 2017). Covering options target exploration more directly by constructing options that reduce expected cover time, linking option discovery to spectral quantities such as algebraic connectivity (Jinnai et al., 2019). Successor Options take a related but distinct approach: instead of using Laplacian eigenvectors, they use successor-based landmark states and define option rewards from changes in discounted future occupancy, yielding options that navigate toward representative regions of the state space (Ramesh et al., 2019). Diffusion options broaden the spectral view by constructing options from the full Laplacian spectrum, allowing both coarse global structure and finer local connectivity patterns to influence the discovered behaviors (Bar et al., 2020). More recently, DCEO showed that Laplacian-based exploratory options can be learned online from pixels in deep RL, substantially extending earlier tabular spectral-option methods (Klissarov & Machado, 2023). Our work is closest to this line, but differs by targeting *novelty-directed* spectral structure rather than a coverage-oriented Laplacian, and by focusing on robust option learning from these rewards.

Novelty-directed exploration. A separate line of work addresses exploration through intrinsic rewards based on novelty or uncertainty. Count-based exploration is the canonical starting point in tabular RL, and pseudo-count methods extended this principle to high-dimensional observations by deriving count-like bonuses from density models (Bellemare et al., 2016; Ostrovski et al., 2017). Random Network Distillation (RND) replaced explicit density estimation with the prediction error of a randomly initialized target network, yielding a simple and scalable novelty bonus that proved highly effective in sparse-reward Atari domains (Burda et al., 2019). These methods can discover underexplored regions, but they generally rely on primitive-action learning and value propagation to translate novelty into purposeful long-horizon behavior. This is precisely the gap that novelty-directed spectral option methods seek to fill.

Episodic exploration. Another important family of intrinsic objectives uses *episodic* novelty rather than purely life-long novelty. Episodic Curiosity measures novelty relative to an episodic memory and uses reachability to avoid the couch-potato failure mode of prediction-error methods (Savinov et al., 2019). Never Give Up and Agent57 combine episodic novelty with life-long novelty, arguing that the two signals play different roles: episodic novelty promotes within-episode behavioral diversity, while life-long novelty preserves pressure toward globally unseen states across training (Badia et al., 2020b;a). E3B also targets episodic exploration, but does so through an uncertainty-based bonus in a learned latent space, encouraging directed exploration of states that are novel within the current episode (Henaff et al., 2022). Together, these works suggest that episodic and life-long novelty provide distinct exploration biases and can be fruitfully combined. Extending NEO-style novelty-directed spectral options with episodic bonuses is a natural direction for future work.

Hybrid intrinsic exploration. Recent work has increasingly explored hybrid intrinsic objectives that combine complementary exploration and exploitation signals. A representative example is

DuRND, which uses dual random-network modules to jointly compute a novelty reward and a contribution-oriented reward, explicitly aiming to balance directed exploration with task progress (Yang et al., 2024). More broadly, such methods are motivated by the observation that a purely novelty-based signal can be insufficiently directional or too myopic, while a more structured intrinsic objective can improve stability and credit assignment in sparse-reward problems. Our reliable NEO formulation fits naturally within this trend, but instantiates the hybridization at the level of *spectral option rewards*, combining eigen-potential directionality with RND-based reliability. More general hybrid intrinsic objectives that additionally incorporate exploitation-oriented signals appear readily compatible with NEO-style option learning, and we leave such combinations to future work.

B Multistep operator preserves eigenvectors

We justify the multistep construction used in Section 3. Let P denote the one-step transition operator under the behaviour policy. For a function $f : \mathcal{S} \to \mathbb{R}$, the one-step operator acts as

$$(Pf)(s) = \mathbb{E}[f(s_{t+1}) \mid s_t = s].$$

A multistep transition operator that compares an anchor state to future states at horizons $1, \ldots, n$ can be written as a weighted sum of powers of P:

$$P_\alpha = \sum_{k=1}^{n} \alpha_k P^k, \qquad \sum_{k=1}^{n} \alpha_k = 1, \qquad \alpha_k \geq 0.$$

Equivalently, P_α is a polynomial in the one-step operator P:

$$P_\alpha = p(P), \qquad p(x) = \sum_{k=1}^{n} \alpha_k x^k.$$

Now suppose that v is an eigenvector of P with eigenvalue λ, so that

$$Pv = \lambda v.$$

Then, for any integer $k \geq 1$,

$$P^k v = \lambda^k v.$$

Therefore,

$$P_\alpha v = \sum_{k=1}^{n} \alpha_k P^k v = \sum_{k=1}^{n} \alpha_k \lambda^k v = \left(\sum_{k=1}^{n} \alpha_k \lambda^k \right) v = p(\lambda) v.$$

Thus, v is also an eigenvector of the multistep operator P_α, with transformed eigenvalue

$$\lambda_\alpha = p(\lambda) = \sum_{k=1}^{n} \alpha_k \lambda^k.$$

This shows that the multistep operator preserves the eigenspaces of the underlying one-step operator while changing only the eigenvalues. Consequently, replacing a one-step smoothness objective by a multistep anchored smoothness objective does not change the spectral directions being targeted in the idealized linear/operator setting. Instead, it reweights the spectrum through the polynomial p. This provides a principled justification for using multistep future states in the representation loss: the objective continues to target the same eigenfunctions, while potentially improving optimization by exposing the representation learner to longer-range connectivity information from sampled trajectories.

In our implementation, the weights $\alpha_{i,k}$ are sample-dependent because they also depend on future-state novelty. The argument above therefore applies exactly when the weights are fixed for a given operator. We use it as the motivating idealization for the multistep construction: the method retains the same spectral intuition as the one-step objective, while adapting the empirical loss to emphasize informative, novel future transitions.

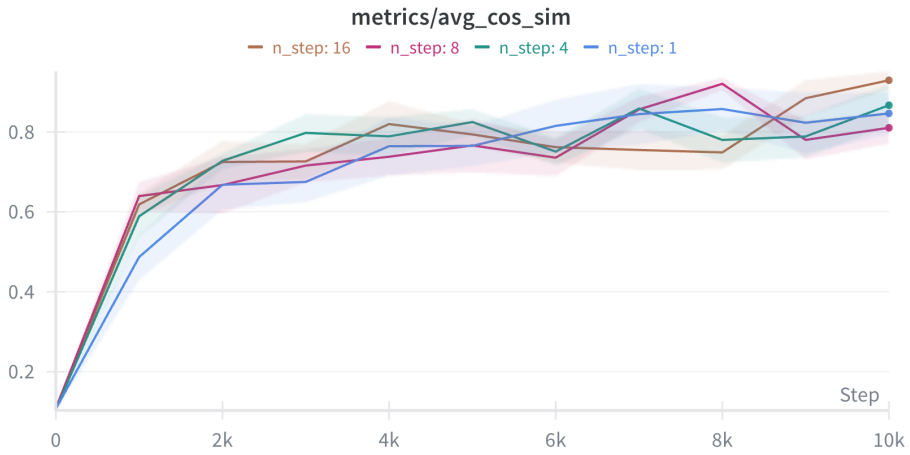


Figure 5: **Multistep graph drawing improves alignment with ground-truth eigenvectors.** Average cosine similarity between the learned and ground-truth eigenvectors for different values of n. Multistep objectives with $n > 1$ reach higher similarity much faster than the one-step objective, indicating more accurate and stable recovery of the target spectral structure.

C Directed kernels, reversibilization, and the usefulness of ALLO

In this appendix we make explicit the reversibilization implicit in Wu-style graph drawing objectives and explain its relation to Chung's directed Laplacian. The purpose of this section is not to claim that the original directed dynamics are reversible, but rather to show that the spectral object optimized by GDO/ALLO is a self-adjoint surrogate obtained from the directed Markov chain.

Setup. Let P be the transition kernel induced by a fixed behaviour policy, and let ρ be a stationary distribution satisfying

$$\rho(v) = \sum_u \rho(u)P(u,v). \tag{11}$$

Let $\Phi = \mathrm{diag}(\rho)$ in the finite-state case. Wu et al. define a symmetric affinity kernel $D(u,v)$ by

$$D(u,v) = \frac{1}{2}\frac{dP(\cdot \mid u)}{d\rho}(v) + \frac{1}{2}\frac{dP(\cdot \mid v)}{d\rho}(u), \tag{12}$$

and the Laplacian operator as

$$\widetilde{L}f = f - \widetilde{D}f. \tag{13}$$

By construction, $D(u,v) = D(v,u)$, so $\widetilde{D}$ is self-adjoint in the weighted Hilbert space $L^2(\rho)$.

C.1 The hidden reverse kernel

In finite state spaces, the Radon–Nikodym derivatives in (12) become

$$D(u,v) = \frac{1}{2}\frac{P(u,v)}{\rho(v)} + \frac{1}{2}\frac{P(v,u)}{\rho(u)}. \tag{14}$$

Multiplying by $\rho(v)$ inside the operator gives

$$(\widetilde{D}f)(u) = \sum_v D(u,v)f(v)\rho(v) \tag{15}$$

$$= \frac{1}{2}\sum_v P(u,v)f(v) + \frac{1}{2}\sum_v \frac{\rho(v)P(v,u)}{\rho(u)}f(v). \tag{16}$$

Define the ρ-time-reversed kernel

$$P^{\dagger}(u,v) = \frac{\rho(v)P(v,u)}{\rho(u)}. \tag{17}$$

Then (16) becomes

$$(\widetilde{D}f)(u) = \frac{1}{2}\sum_{v}\Big(P(u,v) + P^{\dagger}(u,v)\Big)f(v). \tag{18}$$

Therefore, Wu's affinity operator is exactly the average of the forward kernel and the stationary time-reversal kernel:

$$\widetilde{D} = \frac{1}{2}(P + P^{\dagger}), \qquad \widetilde{L} = I - \frac{1}{2}(P + P^{\dagger}), \tag{19}$$

where these equalities are understood as operator equalities on $L^2(\rho)$.

This makes the reversibilization explicit: although Wu et al. do not phrase the construction in terms of reversed chains, the symmetric kernel they define already averages each forward transition with its stationary reverse counterpart.

C.2 Euclidean matrix form

The weighted-space operator in (19) can be written in standard matrix notation. Since

$$P^{\dagger} = \Phi^{-1}P^{\top}\Phi, \tag{20}$$

the Euclidean matrix corresponding to $\widetilde{L}$ is

$$L = I - \frac{1}{2}\Big(P + \Phi^{-1}P^{\top}\Phi\Big). \tag{21}$$

This is the form emphasized in more recent clarifications of the RL Laplacian literature, precisely to avoid the confusion that arises when Wu's weighted-space operator is read as an ordinary Euclidean matrix.

Moreover, L is Φ-self-adjoint:

$$\Phi L = L^{\top}\Phi. \tag{22}$$

Indeed,

$$\Phi\Big(I - \tfrac{1}{2}(P + \Phi^{-1}P^{\top}\Phi)\Big) = \Phi - \tfrac{1}{2}(\Phi P + P^{\top}\Phi),$$

which is symmetric.

Thus, even though P is directed and non-normal in general, the operator optimized by Wu-style graph drawing is self-adjoint in the appropriate weighted geometry.

C.3 Relation to Chung's directed Laplacian

Chung defines the normalized directed Laplacian as

$$L_{\mathrm{Ch}} = I - \frac{1}{2}\Big(\Phi^{1/2}P\Phi^{-1/2} + \Phi^{-1/2}P^{\top}\Phi^{1/2}\Big), \tag{23}$$

which is Hermitian in Euclidean coordinates.

The two formulations are related by a similarity transform:

$$L_{\mathrm{Ch}} = \Phi^{1/2}L\Phi^{-1/2}. \tag{24}$$

Indeed,

$$\Phi^{1/2}L\Phi^{-1/2} = \Phi^{1/2}\Big(I - \tfrac{1}{2}(P + \Phi^{-1}P^{\top}\Phi)\Big)\Phi^{-1/2} \tag{25}$$

$$= I - \frac{1}{2}\Big(\Phi^{1/2}P\Phi^{-1/2} + \Phi^{-1/2}P^{\top}\Phi^{1/2}\Big), \tag{26}$$

which is exactly (23).

Therefore:

- Wu's operator in $L^2(\rho)$,
- its Euclidean matrix form L in (21),
- and Chung's normalized directed Laplacian L_{Ch}

all represent the same reversibilized spectral object, expressed in different coordinates.

C.4 Why this matters for ALLO and deep RL

ALLO is formulated for symmetric Laplacian operators and is designed to robustly recover their eigenfunctions under function approximation. When applied to RL data through Wu-style graph drawing, the relevant operator is not the raw directed kernel P, but the self-adjoint surrogate in (19). This explains why ALLO-style methods can still produce useful spectral representations in directed environments: they are optimizing a valid symmetric operator induced by the directed data, rather than incorrectly assuming the original directed chain itself is symmetric.

This observation is limited but important. It does *not* mean that all directional information is preserved, nor that one recovers the exact non-reversible spectrum of P. Rather, the learned representation reflects the geometry of the reversibilized flow induced by the behavior policy and its stationary distribution. In our view, this is the most plausible reason why ALLO-like methods, and by extension our NEO-ALLO variant, remain empirically useful in directed and non-stationary environments such as Atari and Procgen.

Takeaway. The usefulness of ALLO in directed environments comes from an implicit reversibility trick already present in Wu's kernel construction. The graph-drawing objective learns the eigenfunctions of a stationary-flow symmetrization of the directed Markov chain, and this symmetrized operator is equivalent, up to a change of coordinates, to Chung's directed Laplacian.

Interpretation. The important point is that Wu-style graph drawing does not learn the eigenvectors of the raw directed transition matrix. Instead, it learns the eigenvectors of a stationary-flow symmetrization of that matrix. This is precisely the sense in which the method "reverses the kernel": it mixes forward transition flow with reverse transition flow under the stationary distribution. This makes the learned object spectral and stable, while still retaining useful connectivity information from the directed dynamics.

D Online mean calibration of source weights.

Let $\mathrm{nov}(s)$ denote the normalized novelty bonus and define the inverse-count proxy

$$\mathrm{inv}(s) = \frac{1}{\mathrm{nov}(s) + \varepsilon}. \tag{27}$$

Let m_t denote a running estimate of the maximum inverse count at training step t. We first normalize the inverse-count proxy by this running maximum,

$$x_t(s) = \frac{\mathrm{inv}(s)}{\max(m_t, \varepsilon)}, \qquad x_t(s) \in [0, 1], \tag{28}$$

and apply the NEO scaling rule

$$z_t(s) = \delta\, x_t(s)^{1/k}. \tag{29}$$

Here $\delta > 0$ sets the maximal scale and $k > 0$ controls the compression of the inverse-count range.

To keep the overall strength of the source term stable over training, we rescale these raw weights so that their average remains near a prescribed target mean $\bar{\zeta}_{\mathrm{tar}}$. Let

$$\bar{z}_t = \frac{1}{|\mathcal{B}_t|} \sum_{s \in \mathcal{B}_t} z_t(s) \tag{30}$$

denote the average raw weight over the current rollout or batch $\mathcal{B}_t$. The multiplicative rescaling that would match the target mean exactly is

$$a_t^\star = \frac{\bar{\zeta}_{\text{tar}}}{\bar{z}_t + \varepsilon}. \tag{31}$$

Rather than applying this rescaling abruptly, we maintain an exponential moving average scale parameter a_t:

$$a_{t+1} = (1 - \tau_{\text{mean}})\, a_t + \tau_{\text{mean}}\, a_t^\star, \tag{32}$$

where $\tau_{\text{mean}} \in (0, 1]$ is a small update rate. The final calibrated source weights are then

$$\zeta_t(s) = \text{clip}\big(z_t(s)\, a_{t+1},\ 0,\ \delta\big). \tag{33}$$

Equations (31)–(33) ensure that the average source weight stays near the target scale throughout training, while preserving the relative ordering induced by the raw inverse-count transformation in Eq. (29). In this way, familiar states continue to receive larger source weights than novel states, but the global magnitude of the source penalty remains stable even as the novelty distribution changes over time.

E Algorithms

Algorithms 1 and 2 summarize the non-hierarchical and hierarchical versions of our method. Both share the same three online learning components: (i) option learning with the reliable novelty-directed reward, (ii) novelty-weighted source-Laplacian representation learning, and (iii) primitive-action learning with PQN on extrinsic reward. The hierarchical version additionally maintains a persistent meta-buffer of completed option/primitive segments and trains a meta-controller with SMDP TD-λ updates.

F Mechanistic diagnostic against the exact count-based source Laplacian

To test whether learned options follow the intended source-Laplacian direction, we ran an additional diagnostic on GRIDROOM16 and GRIDMAZE19. We trained an online DQN agent with novelty-weighted ALLO/source-Laplacian representation learning for 50k environment steps and saved checkpoints every 5k steps. At each checkpoint, we froze the learned representation and compared the option behavior induced by vanilla NEO and SNEO with that induced by the *exact* tabular source Laplacian built from true visitation counts.

For each checkpoint, we enumerated all valid grid states, tracked tabular visitation counts $N(s)$ from the actual training trajectory, and constructed the exact count-based source Laplacian. We then computed its eigenvectors exactly. Learned dimensions were compared to true dimensions by fixed index, and for each start state and evaluated dimension, we computed the greedy endpoint induced by the true source-Laplacian option as well as the endpoints induced by the learned NEO and SNEO options.

Our diagnostic metric is the Manhattan (ℓ_1) distance between the endpoint of a learned option rollout and the endpoint of the exact greedy rollout under the true source-Laplacian option. Lower values indicate that the learned option more closely matches the behavior prescribed by the exact count-based source Laplacian.

Figure 6 shows this endpoint ℓ_1 distance over training for GRIDROOM16 and GRIDMAZE19. In both environments, SNEO remains consistently closer to the true greedy endpoint than vanilla NEO across checkpoints. The gap is especially clear on GRIDROOM16 (Figure 6a), where SNEO is typically about 1 to 2 ℓ_1 units closer throughout training. The same qualitative pattern appears on GRIDMAZE19 (Figure 6b), though at a smaller absolute scale. These results suggest that the reliable bonus helps learned options more faithfully realize the control direction implied by the exact count-based source Laplacian.

Algorithm 1: Scalable Novel Eigen Options (SNEO)

```
for i = 1 to T do
    if i = 1 or ⊥ then
        τ ← True ;                                  // option termination
        o ← −1 ;                                    // no active option
        d ← 0 ;                                     // current option duration
        Reset environment and observe state s;
    if τ then
        if U(0,1) < ϵ then
            if U(0,1) < μ then
                o ← U(O);
                τ ← False;
                d ← 0;
                a ~ π_o(· | s);
            else
                o ← −1;
                τ ← True;
                a ~ U(A);
        else
            a ← arg max_{a∈A} Q(s,a);
    else
        a ~ π_o(· | s);
    Execute a, observe r, s′, and ⊥ ;               // ⊥ is episode termination
    Store primitive transition (s,a,r,s′) in primitive buffer B;
    if o ≠ −1 then
        d ← d + 1;
        τ ← (d = D) or ⊥;
    s ← s′;
    Sample a minibatch of rollout segments from B;
    Compute RND novelty and calibrated source weights ζ;
    Train each option with the reliable reward in Eq. 10;
    Minimize the novelty-weighted source-Laplacian objective in Eq. 6;
    Train PQN with TD-λ on primitive-buffer extrinsic reward;
```

Figure 7 shows two representative examples from GRIDROOM16 and GRIDMAZE19. In both cases, the reliable SNEO rollout follows the learned eigenvector direction for longer and ends closer to the exact source-Laplacian endpoint than vanilla NEO. These visualizations qualitatively match the endpoint-distance curves in Figure 6: the reliable novelty term helps the learned option better realize the control behavior implied by the exact count-based source Laplacian.

G Sensitivity to the number of options

We also study sensitivity to the number of options by varying the number of learned options in $\{1, 3, 5, 8, 10\}$ and evaluating aggregated performance on the Procgen-4 (Heist, Ninja, Climber, and Jumper) hard environments; the resulting curve is shown in Figure 4. Increasing the number of options substantially improves performance relative to the single-option setting, indicating that multiple novelty-directed behaviors are important for hard exploration Procgen tasks. The largest gains come from moving beyond 1 and 3 options, while performance with 5, 8, and 10 options is very similar. This suggests that the main benefit comes from having several distinct exploratory directions, after which returns largely saturate.

Algorithm 2: Scalable Hierarchical Novel Eigen Options (SHNEO)

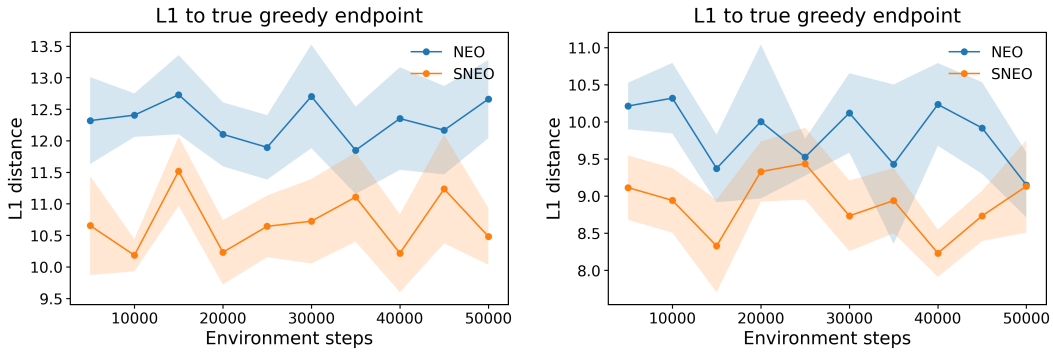
```
for i = 1 to T do
    if i = 1 or ⊥ then
        τ ← True ;                                  // meta-action termination
        u ← −1 ;                                    // no active meta-action
        l ← 0 ;                                     // current meta-action duration
        Reset environment and observe state s;
    if τ then
        if U(0,1) < ϵ then
            if U(0,1) < μ then
                u ← U(O) ;                          // select option
            else
                u ← U(A) ;                          // select primitive action
        else
            u ← arg max_{v∈A∪O} Q_meta(s, v);
        Initialize segment return R ← 0, duration l ← 0, and start state s̄ ← s;
    if u ∈ A then
        a ← u;
        τ ← True;
    else
        a ~ π_u(· | s);
    Execute a, observe r, s′, and ⊥;
    Store primitive transition (s, a, r, s′) in primitive buffer B;
    Update segment return R ← R + γ^l r and duration l ← l + 1;
    if u ∈ O then
        τ ← (l = D) or ⊥;
    if τ or ⊥ then
        Store completed meta-transition (s̄, u, R, l, s′) in persistent meta-buffer B̄;
    s ← s′;
    Sample a minibatch of rollout segments from B;
    Compute RND novelty and calibrated source weights ζ;
    Train each option with the reliable reward in Eq. 10;
    Minimize the novelty-weighted source-Laplacian objective in Eq. 6;
    Train PQN with TD-λ on primitive-buffer extrinsic reward;
    Sample a minibatch of completed meta-transitions from B̄;
    Train the meta-controller with SMDP TD-λ targets;
```

H Experimental Details

This appendix provides the implementation and tuning details for all experiments reported in the main paper. Unless otherwise noted, all methods use the same PQN backbone and differ only in the additional exploration, spectral representation, option learning, or hierarchical control components.

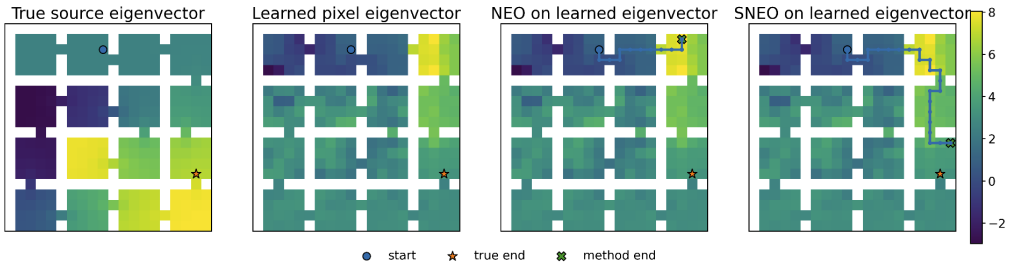
H.1 Common training setup

Across all methods, the backbone learner is PQN with the standard convolutional encoder (Mnih et al., 2013) used in our Atari and Procgen experiments. For option-based methods, we use a shared convolutional encoder together with separate feed-forward Q-heads for each option. Unless otherwise stated, all option-based methods use 10 options. The option duration is fixed to 5 in Procgen and 10 in Atari. Procgen results are trained for 25M environment steps and Atari results for 50M environment

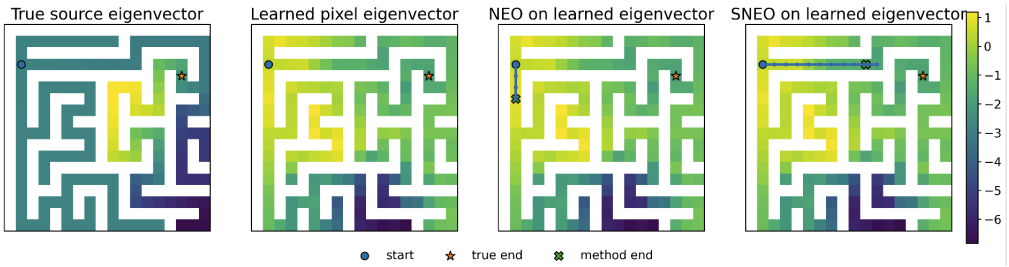


(a) GRIDROOM16 (b) GRIDMAZE19

Figure 6: **Endpoint distance to the exact source-Laplacian option.** Mean Manhattan (ℓ_1) distance between the endpoint of the learned option rollout and the endpoint of the exact greedy rollout induced by the true count-based source Laplacian, averaged over evaluated dimensions, valid start states, and seeds. Lower is better. In both environments, SNEO remains consistently closer to the exact source-Laplacian endpoint than vanilla NEO throughout training.



(a) GRIDROOM16



(b) GRIDMAZE19

Figure 7: **Example option rollouts under the exact count-based source-Laplacian diagnostic.** Each panel shows, from left to right, the exact source-Laplacian eigenvector, the learned pixel eigenvector, the rollout induced by vanilla NEO on the learned eigenvector, and the rollout induced by SNEO on the learned eigenvector. The blue circle marks the start state, the orange star marks the endpoint of the exact source-Laplacian option, and the green cross marks the endpoint of the learned option. In both examples, the SNEO rollout follows the intended source-Laplacian direction for longer and terminates closer to the exact endpoint than vanilla NEO.

steps, except for MONTEZUMA'S REVENGE, which is trained for 1B environment steps. Results are averaged across 5 seeds, except for the 1B-step Montezuma runs, which are averaged across 3 seeds.

We evaluate Procgen in the same-level (seed 0) setting under the hard distribution, since our objective is to study exploration rather than procedural generalization. The Procgen Hard-7 subset consists of BOSSFIGHT, CLIMBER, FRUITBOT, HEIST, JUMPER, LEAPER, and NINJA. For Atari, we report results on MONTEZUMA'S REVENGE and VENTURE.

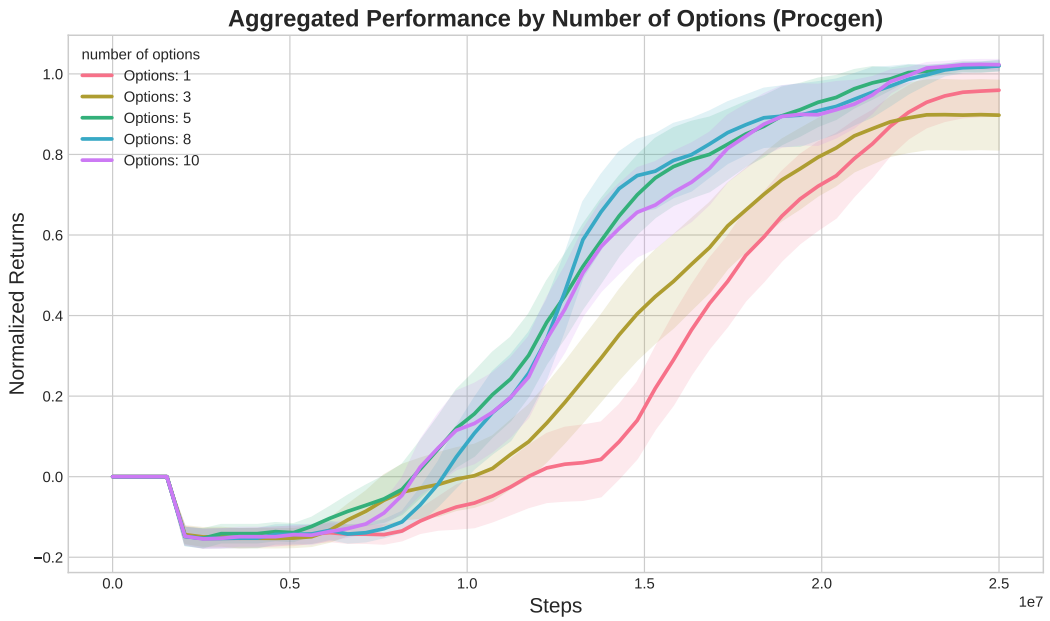


Figure 8: **Sensitivity to the number of options.** Aggregated normalized return on the Procgen hard environments for $\{1, 3, 5, 8, 10\}$ learned options. Performance improves substantially over the single-option setting, while 5, 8, and 10 options perform similarly, suggesting that returns largely saturate once a sufficiently diverse set of exploratory directions is available.

H.2 Compared methods

We compare the following methods:

- **PQN**: vanilla PQN without intrinsic exploration.
- **RND**: PQN with Random Network Distillation intrinsic bonus.
- **EO**: PQN with Deep Covering Eigen Options.
- **HEO**: hierarchical version of EO.
- **NEO**: deep novelty-directed eigenoptions without reliability augmentation.
- **HNEO**: hierarchical version of NEO.
- **R:NEO**: reliable NEO, which augments eigen-potential rewards with the RND bonus.
- **R:HNEO**: hierarchical reliable NEO.

For fair comparison, all methods use the same environment wrappers, rollout budget, encoder family, and optimization backbone unless the method itself requires an additional component.

H.3 Shared backbone hyperparameters

The following are backbone PQN hyperparameters; we retain most of the settings from their paper (Gallici et al., 2025):

- Discount factor γ: `[0.99]`
- Number of parallel environments: `[128]`
- Rollout length T: `[32]`

- Number of epochs per PQN update: [2]
- Number of minibatches: [32]
- Optimizer: [AdamW (Loshchilov & Hutter, 2017)]
- Learning rate: [0.00025
- Epsilon-greedy exploration schedule: [Epsilon-end: 0.1, Epsilon-fraction: 0.1]
- Network architecture details: [ATARI DQN (Mnih et al., 2013)]
- layer normalization choices: [Yes]
- TD-λ: [0.5, 0.65, 0.8, 0.95]

For ATARI, we used TD-λ of 0.65 as recommended in the PQN paper, but for Procgen, we used 0.95, the highest value that gave the best aggregate test performance.

H.4 RND hyperparameters

For all methods using RND, we use the same predictor-target setup and normalization pipeline. The hyperparameters swept, and the final selected values in bold were:

Hyperparameters considered.

- Intrinsic reward coefficient: [0.1, 0.5, 1.0]
- RND predictor learning rate: [0.00025]
- Observation normalization / burn-in length: [RMS Normalization (Yuan et al., 2024) and 500_000 burin-in steps]
- Reward normalization settings: [RMS Normalization (Yuan et al., 2024)]
- Intrinsic discount factor: [0.99]
- RND update proportion: [0.1, 0.25, 1.0]

Finally, we used an intrinsic reward coefficient of 0.5 in ATARI and 1.0 in Procgen, while the update proportion was 0.25, chosen based on the best test performance within the ATARI and Procgen suites combined.

H.5 Laplacian / ALLO hyperparameters

All spectral methods use the same Laplacian representation-learning module unless noted otherwise. Our representation objective combines a multistep anchored smoothness term, a source-Laplacian sink term, and ALLO-style orthonormality control.

Hyperparameters considered.

- Laplacian embedding dimension: [32]
- Number of future steps in multistep ALLO: [32 (rollout length)]
- Barrier coefficient learning rate: [0.01]
- Initial Barrier Value: [0.1, 1, 10, 100, 512, 3096]
- Dual variable learning rate: [0.01]

- Error-moving-average update rate: `[0.01]`
- batch normalization in the Laplacian network: `[Yes]`
- Laplacian network learning rate: `[0.00025]`

The most important hyperparameter for Laplacian Optimization is the initial barrier coefficient. We empirically found that a higher initial value enforces the orthonormal constraint more strictly from the start, leading to faster convergence in Tabular Grid world environments with pixel data. So we extended the same to pixel-based Atari and Procgen, using a value of 100 for all vanilla Laplacian-embedding-based models. For NEO and its variants, incorporating count information into the diagonal entries of the Laplacian required a much higher initial barrier coefficient to enforce a strict orthonormal constraint; hence, we chose 3096.

H.6 NEO scaling and novelty-to-ζ conversion

For NEO-style methods, we convert normalized RND novelty into inverse-count proxies and then into source-Laplacian weights ζ. The source scaling uses the calibrated NEO transformation described in Appendix D.

Hyperparameters used.

- Source-weight scale δ: `[0.5]`
- Source-weight exponent k: `[1]`
- Target mean for online ζ calibration: `[0.3]`
- EMA update rate for running maximum: `[0.001]`
- EMA update rate for mean calibration: `[0.001]`

H.7 Option-learning hyperparameters

All option-based methods use separate option-value heads and fixed-duration execution. For vanilla EO/HEO and NEO/HNEO, option rewards are given by the corresponding eigen-potential differences. For reliable variants, the option reward additionally includes the RND-based reliability bonus.

Hyperparameters considered.

- Number of options: `[1, 5, 10]`
- Option duration: `[5, 10]`
- Option-policy learning rate: `[0.00025]`
- Option-policy TD-λ: `[same as main policy]`
- Option-policy discount factor: `[0.95]`
- Option-policy exploration temperature at train: `[0.25]`

We fixed the number of options at 10, after which we observed the results to saturate. In fact, the improvement from 5 options itself was minimal in many environments. We used option durations of 10 in Atari, following previous work (Klissarov & Machado, 2023), and in Procgen we tried 5 and 10, finding that a shorter duration worked better in a stable sense.

H.8 Hierarchical controller hyperparameters

Hierarchical methods maintain a persistent meta-buffer over completed macro-actions and train the meta-controller with SMDP TD-λ. Primitive actions are treated as length-one options in the ordered meta-action set.

Hyperparameters considered.

- Meta-buffer length M: `[8]`
- Meta-controller TD-λ: `[same as primitive, 0.65 for ATARI and 0.95 for Procgen]`
- Probability of exploring with options versus primitives (μ): `[0.5 and 0.9]`
- Meta-controller learning rate: `[0.00025]`

We resorted to using a smaller value of 8 for the meta-buffer length, as with an option duration of 10, the primitive timestep might grow to as much as 10 times the meta-buffer length. The most important hyperparameter here is μ, the option-probability with we sample an option relative to a primitive action. Following earlier work (Machado et al., 2023), we used 0.9 in ATARI and we tested 0.5 and 0.9 in Procgen and then finally used 0.5 at the end. This value depends on the sparsity level. Procgen had relatively denser rewards than Montezuma and Venture; hence, it benefited equally from sampling primitives and options.

H.9 Reliable NEO hyperparameters

Reliable NEO augments the eigen-potential option reward with the RND bonus. This introduces one main additional coefficient.

Hyperparameters considered.

- Reliability coefficient β_{rnd} / `rnd_coef`: `[1, 10, 100]`

While there was little difference between 10 and 100, we used 100. for the reliability coefficient.

H.10 Model selection and tuning protocol

Hyperparameters were tuned on aggregate scores in Procgen-4 and in Montezuma (50 M budget) and then fixed across the reported benchmark suite. Selection was based on sample efficiency within the corresponding training budget.

Concretely, we followed the protocol below:

1. Start from a common PQN backbone that performs reasonably across the target domain (TD-λ).
2. Tune the RND baseline first to obtain a strong global novelty signal and achieve literature-comparable performances in the environment suites (Update Proportion, reward coefficient, and RMS normalization).
3. Add the Laplacian / ALLO representation layer and tune its stability-sensitive hyperparameters (barrier value).
4. Tune option-learning hyperparameters for EO/NEO-style methods (number of options, option duration, discount factor).
5. Finally, tune the reliability coefficient and hierarchical controller settings for R:HNEO (β, barrier value).

H.11 Environment-specific notes

Procgen. All Procgen experiments are run in same-level evaluation mode under the hard distribution. We selected the Hard-7 subset by first running vanilla PQN on all Procgen tasks and retaining those environments where the normalized score failed to reach 1 within the allotted budget.

Atari. We report results on MONTEZUMA'S REVENGE and VENTURE. MONTEZUMA'S REVENGE is trained for 1B steps because even strong novelty-driven baselines remain highly sparse over much of training. VENTURE is trained for 50M steps. Additional hard games such as PITFALL! and PRIVATE EYE were not included in the final benchmark suite for the reasons discussed in Section 6.

H.12 Reproducibility

For each reported setting, we fix the random seed and average over multiple runs. Procgen and standard Atari results are averaged over 5 seeds, while the 1B-step Montezuma runs are averaged over 3 seeds due to computational cost. We report normalized returns in Procgen and raw environment returns in Atari. All code, configuration files, and exact command lines will be released upon publication.

H.13 Compute Resources and Compute Time

The experiments were run on an Nvidia L40S GPU and 32 CPUs in a cluster, which was sufficient. Our most computationally intensive model is the HNEO family. Under this model, running 5 experiments (seeds) together took 1.15 hrs per Progen game over 25 million steps, and under 4 hours for Atari Venture. Running 3 seeds in Montezuma Revenge took under 2days and 2 hours per game.