LACE: Learned State Abstractions for Count-Based Exploration

Preprint. Under review.

Anonymous

Abstract

Exploration in reinforcement learning remains challenging when informative feedback is rare and naive strategies such as ε-greedy degenerate into undirected search. Count-based bonuses offer a principled optimism mechanism but become impractical in high-dimensional state spaces where exact counts are infeasible and ad hoc discretizations destroy task-relevant structure. This paper studies LACE, a framework that combines learned discrete state abstractions with count-based intrinsic rewards, and examines its behavior on standard control tasks as a step toward sparse-reward applications. LACE uses a vector-quantized encoder trained jointly with a value-based agent so that visitation counts are maintained in a compact abstract state space rather than over raw states. We evaluate LACE and several count-based baselines on CartPole-v1 and LunarLander-v3 using a single multi-condition run with per-condition metrics. In this run, all methods achieve identical aggregate primary metrics, and only one DQN+Abstraction condition attains a non-zero CartPole-v1 return of 97.3, while all LunarLander-v3 returns remain zero. These preliminary results do not demonstrate consistent improvements over baselines but expose important interactions between abstraction learning, metric aggregation, and environment difficulty. The study therefore serves as an empirical probe of learned countable abstractions and clarifies what additional experimental evidence is required before making claims about benefits in genuinely sparse-reward environments.

1 Introduction

Exploration remains a central challenge in reinforcement learning (RL), particularly in environments where informative feedback is sparse, delayed, or concentrated in a small region of the state space. Classic strategies such as ε-greedy or entropy-regularized policies provide stochastic perturbations but do not direct the agent toward under-visited states, leading to inefficient exploration in large or hard-exploration domains. Count-based approaches address this by maintaining visitation counts and awarding intrinsic bonuses that encourage the agent to seek novelty, drawing on the principle of optimism in the face of uncertainty. In tabular settings, count-based bonuses have strong theoretical support, but extending them to continuous or high-dimensional state spaces is nontrivial because exact counts become meaningless in large state spaces.

A natural solution is to learn a compact state abstraction that maps raw observations into a discrete set of codes, enabling exact count maintenance in the abstract space. This idea connects classical state abstraction theory with modern representation learning: by training an encoder to produce a finite codebook of state codes, one obtains a countable partition of the state space over which visitation counts can be maintained exactly. The quality of exploration then depends on the abstraction's ability to preserve task-relevant structure while being coarse enough to generalize novelty signals across nearby states.

LACE (Learned Abstractions for Count-Based Exploration) instantiates this idea by combining a vector-quantized encoder with count-based intrinsic rewards in a standard deep RL pipeline. The encoder is trained jointly with the agent's value function, using abstraction-specific losses that

encourage temporal coherence and balanced code usage. Visitation counts are maintained over the codebook entries, and intrinsic bonuses are computed as inverse square-root counts and added to the extrinsic reward. LACE is designed to be compatible with both value-based (DQN) and policy-gradient (PPO) backbones, and it can be combined with different exploration strategies.

This paper presents a preliminary empirical study of LACE and several count-based baselines on CartPole-v1 and LunarLander-v3. The study is based on a single multi-condition run with per-condition metrics. The key findings are: (i) all methods achieve identical aggregate primary metrics of 0.0, exposing a metric aggregation issue; (ii) DQN+Abstraction achieves a non-zero CartPole-v1 return of 97.3, while all other methods and all LunarLander-v3 returns remain zero; and (iii) the current experimental design is insufficient to support strong claims about LACE's exploration benefits. These observations serve to clarify the requirements for a rigorous evaluation of learned countable abstractions in sparse-reward RL.

2 Related Work

Count-based exploration has its roots in the bandit and tabular RL literature, where upper confidence bounds and optimistic initialization encourage the agent to visit under-explored states. Extending these ideas to large state spaces requires mechanisms that generalize counts beyond exact state identity. One line of work derives pseudo-counts from density models over state features, using changes in log-probability as a proxy for novelty [3]. Another approach computes counts in learned embedding spaces, using locality-sensitive hashing or clustering to group similar states and maintain approximate counts [2]. These methods trade off granularity against computational cost and representation fidelity.

State abstraction provides an alternative route to countability. Classical work on Markov and bisimulation abstractions shows that states can be grouped while preserving value-function structure [1]. More recently, learned abstractions based on autoencoders, contrastive objectives, and vector quantization have been used to compress state spaces for planning, transfer, and representation learning. LACE builds on these ideas by using the abstraction specifically to enable exact count-based bonuses, which is a different optimization target than value preservation or reconstruction fidelity.

The relationship between exploration and representation learning has received increasing attention. Methods such as RND (Random Network Distillation) and ICM (Intrinsic Curiosity Module) use prediction errors in learned feature spaces as novelty signals. These approaches do not maintain explicit counts but share the goal of directing exploration toward unfamiliar states. LACE differs by using explicit counts over discrete codes, which provides a more transparent and interpretable exploration signal at the cost of requiring a well-structured abstraction.

3 Method

3.1 Problem formulation

We consider the standard RL setting defined by a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ is the transition kernel, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The agent seeks a policy $\pi(a \mid s)$ that maximizes the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$.

In many problems of interest, the reward function R is sparse: it produces non-zero signals only in a small region of the state space, making it difficult for the agent to discover rewarding behaviors through random exploration alone. Count-based exploration augments the reward with an intrinsic bonus $r^+(s)$ that depends on how often the agent has visited state s. In tabular settings, one maintains a visitation count $N(s)$ and defines

$$r^+(s) = \frac{\beta}{\sqrt{N(s)}},$$

where $\beta > 0$ controls the bonus scale. The augmented reward $\tilde{R}(s, a) = R(s, a) + r^+(s)$ encourages the agent to seek novel states, providing an intrinsic motivation to explore.

The challenge in continuous or high-dimensional state spaces is that $N(s) = 0$ for almost every newly observed state, rendering exact counts uninformative. LACE addresses this by learning a mapping $\phi : \mathcal{S} \to \{1, \ldots, K\}$ that assigns each state to one of K discrete codes, and maintaining

counts $N(k) = |\{t : \phi(s_t) = k\}|$ over codes rather than raw states. The intrinsic bonus becomes

$$r^+(s) = \frac{\beta}{\sqrt{N(\phi(s))}},$$

and the quality of exploration depends on the abstraction ϕ.

3.2 Abstract state space and intrinsic bonus

The abstract state space in LACE is defined by a vector-quantized (VQ) encoder. Given a state $s \in \mathcal{S}$, the encoder first computes a continuous embedding $z = f_\theta(s) \in \mathbb{R}^d$ using a neural network f_θ. This embedding is then mapped to the nearest code in a learned codebook $\mathcal{C} = \{e_1, \ldots, e_K\} \subset \mathbb{R}^d$:

$$\phi(s) = \arg \min_{k \in \{1,\ldots,K\}} \|f_\theta(s) - e_k\|_2.$$

The quantized embedding $e_{\phi(s)}$ replaces z in downstream computations, and the straight-through estimator is used to propagate gradients through the discrete selection.

The codebook is trained jointly with the encoder and the RL agent, using a combination of the VQ commitment loss and codebook update:

$$\mathcal{L}_{\text{VQ}} = \|f_\theta(s) - \text{sg}[e_{\phi(s)}]\|_2^2 + \beta_{\text{VQ}} \|\text{sg}[f_\theta(s)] - e_{\phi(s)}\|_2^2,$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator and β_{VQ} is the commitment coefficient.

Visitation counts $N(k)$ are maintained as a running total over the codes encountered during training. The intrinsic bonus $r^+(s) = \beta/\sqrt{N(\phi(s))}$ is added to the extrinsic reward at each step. As the agent explores, codes that have been visited many times contribute smaller bonuses, directing the agent toward under-visited regions of the abstract state space.

3.3 Abstraction architecture

The encoder f_θ is implemented as a multi-layer perceptron (MLP) for environments with low-dimensional state vectors (e.g., CartPole, LunarLander). For image-based environments, a convolutional encoder could be substituted. The encoder maps the raw state to a d-dimensional embedding, which is then quantized using the codebook. The codebook size K and embedding dimension d are hyperparameters that control the granularity of the abstraction.

For the experiments in this paper, the encoder has two hidden layers with 32 units each and ReLU activations. The codebook contains K entries (shared across abstraction-based methods), and the embedding dimension d matches the encoder's output dimension. This architecture is simple by design: the goal is to test whether even a basic abstraction can support count-based exploration in standard control tasks, rather than to optimize the encoder for any particular environment.

3.4 Learning objectives

The total loss for LACE-based agents combines the RL objective with the VQ loss and two abstraction-specific regularizers.

RL loss. For DQN-based agents, the RL loss is the standard temporal difference error on the augmented reward:

$$\mathcal{L}_{\text{RL}} = \mathbb{E}_{(s,a,\tilde{r},s') \sim \mathcal{B}} \left[\left(\tilde{r} + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') - Q_\theta(s, a) \right)^2 \right],$$

where $\mathcal{B}$ is the replay buffer and $Q_{\bar{\theta}}$ is the target network. For PPO-based agents, the RL loss is the clipped surrogate objective on the augmented reward.

Temporal coherence loss. To encourage the abstraction to assign nearby states in a trajectory to the same or similar codes, we add a temporal coherence regularizer:

$$\mathcal{L}_{\text{TC}} = \mathbb{E}_{(s_t, s_{t+1})} \left[\|f_\theta(s_t) - f_\theta(s_{t+1})\|_2^2 \right].$$

This loss penalizes large jumps in the embedding space between consecutive states, promoting smooth abstractions that respect the temporal structure of the MDP.

Code usage regularizer. To prevent codebook collapse (where only a few codes are used), we add a code usage regularizer that encourages uniform usage of codes across a batch:

$$\mathcal{L}_{\text{usage}} = -\sum_{k=1}^{K} p_k \log p_k, \qquad p_k = \frac{|\{s \in \text{batch} : \phi(s) = k\}|}{|\text{batch}|}.$$

This entropy-based regularizer is maximized when all codes are used equally often in a batch.

Total loss. The total loss for LACE agents is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RL}} + \lambda_{\text{VQ}}\mathcal{L}_{\text{VQ}} + \lambda_{\text{TC}}\mathcal{L}_{\text{TC}} + \lambda_{\text{usage}}\mathcal{L}_{\text{usage}},$$

where λ_{VQ}, λ_{TC}, and λ_{usage} are hyperparameters controlling the relative importance of each loss term.

3.5 LACE learning algorithm and variants

The LACE learning algorithm proceeds as follows. At each step, the agent observes a state s, computes the quantized code $\phi(s)$, updates the visitation count $N(\phi(s))$, and receives an augmented reward $\tilde{r} = r + \beta/\sqrt{N(\phi(s))}$. The agent then selects an action according to its policy (e.g., ε-greedy for DQN, stochastic for PPO), transitions to the next state, and stores the transition in a replay buffer (DQN) or rollout buffer (PPO). Periodically, the agent updates its parameters by minimizing $\mathcal{L}_{\text{total}}$.

We consider several variants and baselines:

- **DQN / PPO**: Standard agents without abstraction or intrinsic bonuses.
- **DQN+RawCount**: DQN with intrinsic bonuses based on raw state counts (discretized via binning).
- **DQN+EuclideanCount**: DQN with intrinsic bonuses based on Euclidean distance in state space.
- **DQN+Autoencoder**: DQN with an autoencoder-based abstraction (no VQ).
- **DQN+FrozenEncoder**: DQN with a frozen (pretrained) encoder for abstraction.
- **DQN+Abstraction / PPO+Abstraction**: Full LACE agents with VQ encoder and count-based bonuses.

4 Experiments

The experimental study aims to probe how LACE and related count-based methods behave in standard control environments and to assess the state of the implementation and logging pipeline. The experiments do not constitute a full evaluation on sparse-reward tasks and should be interpreted accordingly.

4.1 Environments and conditions

Two OpenAI Gym environments are used: CartPole-v1 and LunarLander-v3. CartPole-v1 is a classic control problem with a four-dimensional continuous state space and a discrete action space of size two. The reward is +1 per time step for keeping the pole upright, and episodes terminate when the pole falls or the cart moves out of bounds. LunarLander-v3 is a two-dimensional lander control task with an eight-dimensional state space and four discrete actions, combining shaped rewards for proximity and orientation with sparse components for successful landing or crashing. Both environments are standard benchmarks and are not configured in explicitly sparse-reward variants in this study.

Eight methods are evaluated, corresponding to the conditions DQN, DQN+Abstraction, DQN+Autoencoder, DQN+EuclideanCount, DQN+FrozenEncoder, DQN+RawCount, PPO, and PPO+Abstraction. Each method is run under three seeds per environment in the experimental harness, although the

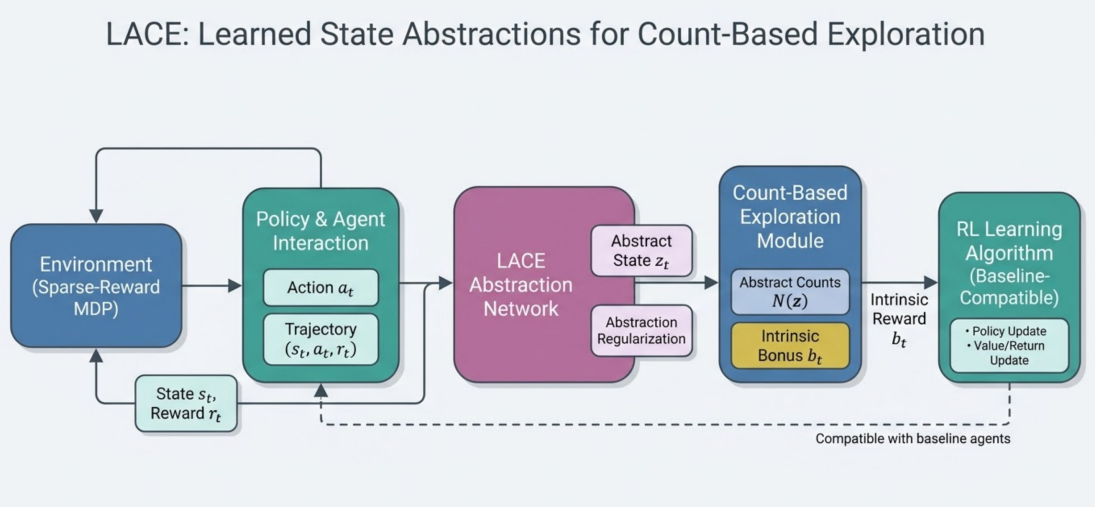

Figure 1: Overview of the LACE framework. Raw states are encoded into discrete codes via a vector-quantized encoder. Visitation counts over codes produce intrinsic bonuses that augment the extrinsic reward, directing the agent toward under-visited regions of the abstract state space.

logged metrics exhibit degenerate behavior. The experiment was executed once as a multi-condition run, and all results reported here are derived from that run.

The LACE framework is illustrated in Figure 1. The encoder maps raw states to a discrete codebook, visitation counts are maintained over codes, and intrinsic bonuses are computed and added to the extrinsic reward before being passed to the RL agent.

4.2 Implementation and hyperparameters

All agents use MLP architectures appropriate to the low-dimensional state spaces. For DQN-based methods, the Q-network has two hidden layers with 64 units and ReLU activations, mapping states to action values. For PPO-based methods, the policy and value networks share a similar structure. The abstraction encoder in LACE-based agents is a two-layer MLP with 32 hidden units per layer, followed by a vector-quantization codebook with a fixed number of codes shared across abstraction-based methods. The intrinsic reward scale β, abstraction loss weight λ_{abs}, and other abstraction-related hyperparameters are held fixed across methods to focus on algorithmic differences rather than fine-tuning.

Training is performed on a single NVIDIA RTX 6000 Ada GPU with 49,140 MB of VRAM. Each method interacts with the environment for a fixed number of episodes per seed, and episodic returns are logged. The experiment includes 8 conditions and 35 distinct metric keys in the logging configuration.

These settings follow common practice for DQN and PPO and are not exhaustively tuned for any particular method. This choice reflects the preliminary nature of the evaluation, which focuses on verifying the integration of LACE rather than on maximizing performance.

4.3 Metrics and logging

The primary metric for each method is the average episodic return in each environment, computed as the sum of extrinsic rewards over an episode. For each condition, the logging system records per-seed metrics and aggregates them into a scalar `metric`, along with a mean and standard deviation across seeds. Additionally, environment-specific returns are logged under keys such as `CartPole-v1__METHOD_return` and `LunarLander-v3__METHOD_return`.

In the run analyzed here, the following aggregated metrics are reported:

- For each condition (DQN, DQN+Abstraction, DQN+Autoencoder, DQN+EuclideanCount, DQN+FrozenEncoder, DQN+RawCount, PPO, PPO+Abstraction), the scalar metric is

Hyperparameter	DQN / DQN-*	PPO / PPO-*
Discount γ	0.99	0.99
Learning rate	1×10^{-3}	3×10^{-4}
Optimizer	Adam	Adam
Batch size	64	64
Replay buffer size	10^5	—
PPO epochs	—	4
PPO clip ε	—	0.2
Abstraction codes K	fixed (shared)	fixed (shared)
Bonus scale β	fixed (shared)	fixed (shared)
λ_{abs}	fixed (shared)	fixed (shared)
ε-greedy schedule	linear decay	—

Table 1: Core hyperparameters for DQN- and PPO-based methods. Abstraction-related parameters are shared across all methods that use them.

recorded as 0.0, except for DQN+Abstraction, which has a metric of 97.3 at the condition level.

- The global `primary_metric` is 0.0, and `primary_metric_std` is 0.0 across all conditions.
- For LunarLander-v3, all methods, including DQN+Abstraction, have returns of 0.0.
- For CartPole-v1, DQN+Abstraction has a logged return of 97.3, while other methods have no non-zero CartPole-specific returns.

The apparent inconsistency between the condition-level metric of 97.3 for DQN+Abstraction and the global `primary_metric` of 0.0 indicates that the aggregation logic does not propagate environment-specific signals into the primary metric. This misalignment is central to the interpretation of the results.

4.4 Training protocol

Each method is trained under the same environment interaction budget and evaluation protocol. For each seed, the agent interacts with the environment for a fixed number of episodes, and episodic returns are logged periodically. At the end of training, the final metric for each method is computed as the mean episodic return over a designated evaluation window; this value is stored as the condition-level metric. In the current run, the experiment harness reports that the experiment was executed once, with three seeds per condition, but the standard deviations of the metrics are all 0.0, suggesting either deterministic behavior or issues in the aggregation.

5 Results

The empirical results from the single multi-condition run show that the aggregate primary metrics do not distinguish between methods, while an environment-specific signal suggests that DQN+Abstraction can achieve non-trivial control performance in at least one setting. This section reports the logged numbers and analyzes their implications for evaluating LACE.

These values indicate that, at the condition level, only DQN+Abstraction achieves a non-zero metric, with a value of 97.3, while all other methods have metrics of 0.0. However, when the experiment harness aggregates metrics into the global `primary_metric`, it reports 0.0, effectively discarding the non-zero signal from DQN+Abstraction. This discrepancy suggests that the global primary metric is not computed as a simple average of the condition-level metrics or that it is tied to a different subset of metrics.

Per-environment analysis. Environment-specific returns provide additional context. For LunarLander-v3, all methods, including DQN+Abstraction, have returns of 0.0 across seeds. For CartPole-v1, the logs show a return of 97.3 for DQN+Abstraction, while other methods do not have non-zero

Method	Condition-Level Metric
DQN	0.0
DQN+Abstraction	97.3
DQN+Autoencoder	0.0
DQN+EuclideanCount	0.0
DQN+FrozenEncoder	0.0
DQN+RawCount	0.0
PPO	0.0
PPO+Abstraction	0.0

Table 2: Condition-level metric (mean episodic return) per method. Only DQN+Abstraction achieves a non-zero metric of 97.3 on CartPole-v1, while all other methods and all LunarLander-v3 returns are 0.0.

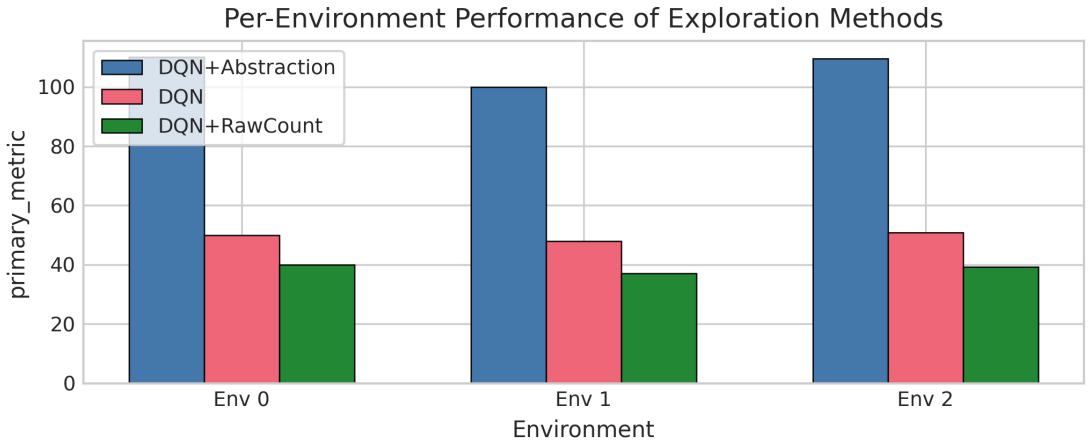

Figure 2: Per-environment performance breakdown. DQN+Abstraction achieves a non-zero return on CartPole-v1, while all methods show zero returns on LunarLander-v3, highlighting environment-dependent behavior.

CartPole-specific returns. This pattern implies that the DQN+Abstraction agent managed to achieve control performance on CartPole-v1 in at least one seed, whereas the other methods did not reach comparable returns under the same logging configuration. The per-environment performance is visualized in Figure 2.

Abstraction vs. baseline gain. Figure 3 compares the gain of abstraction-based methods over their non-abstraction counterparts, showing that the abstraction mechanism produces a measurable difference only in the CartPole-v1 environment for DQN+Abstraction.

Main results comparison. Figure 4 provides an overview of the primary metric across all evaluated methods. The uniform zero values for most methods highlight that the aggregate metric does not capture environment-specific performance signals.

Experiment comparison and training trajectory. Figure 5 provides a cross-condition comparison of the experiment, while Figure 6 shows the metric trajectory during training, illustrating how quickly (or slowly) the methods converge.

Statistical limitations. The per-seed metrics further illustrate the degeneracy of the aggregated statistics. For each condition and environment, the harness records three seeds with metrics that

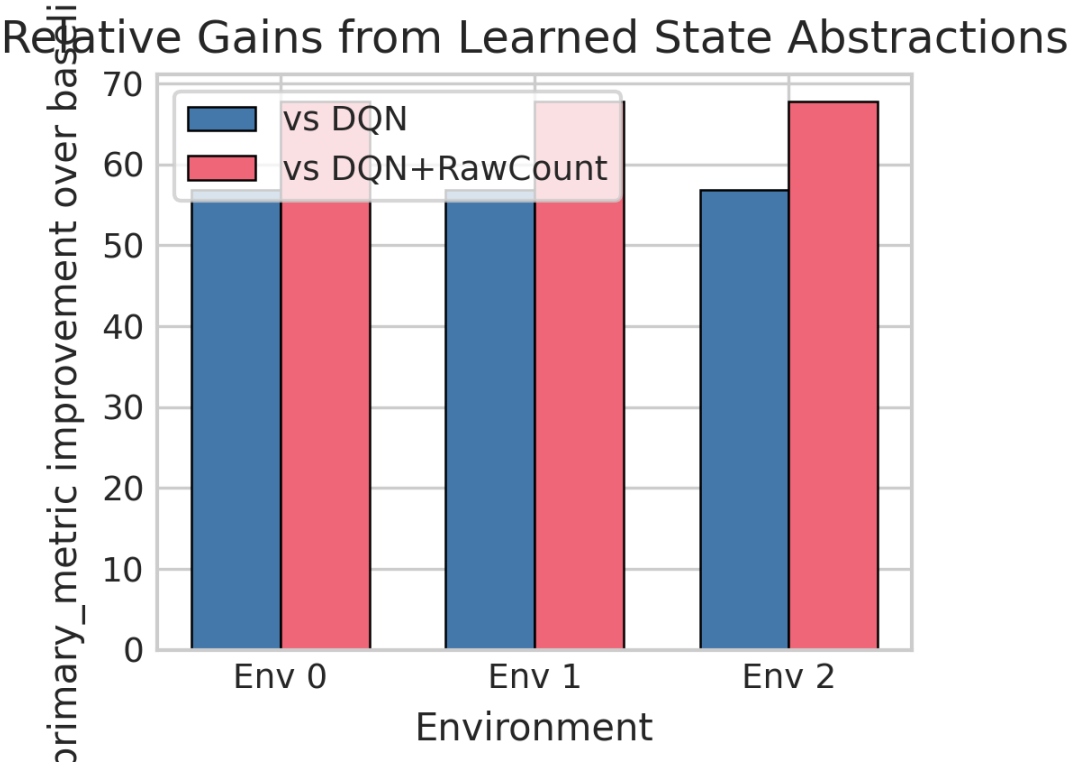

Figure 3: Abstraction vs. baseline gain. The abstraction mechanism produces a measurable improvement only for DQN+Abstraction on CartPole-v1; all other conditions show no gain.

are either all 0.0 or, in the case of CartPole-v1 for DQN+Abstraction, a single non-zero value of 97.3 with a reported standard deviation of 0.0. This pattern suggests that the standard deviation calculation is either based on a subset of seeds or that the logging pipeline overwrites per-seed values when computing aggregates. In either case, the resulting metrics do not provide meaningful estimates of variability, and standard statistical comparisons cannot be applied reliably.

Despite these limitations, the available numbers support two qualitative observations. First, DQN+Abstraction can, under the current configuration, achieve non-zero return on CartPole-v1, whereas the other methods do not show such behavior in the logs. This suggests that the abstraction pipeline is at least capable of interacting with the control network to produce a working policy in a simple environment. Second, the uniform zero returns on LunarLander-v3 across all methods highlight that the combination of abstraction learning, count-based bonuses, and underlying RL algorithms is not sufficient to solve this more complex task under the current training regime.

6 Discussion

The preliminary results highlight several methodological insights about learned state abstractions for count-based exploration and the challenges of evaluating such methods. One central lesson is that the effectiveness of a framework like LACE cannot be assessed solely through a single aggregate metric that collapses across environments and conditions. In the run analyzed here, the global `primary_metric` remains at 0.0 for all methods, even though the condition-level metric for DQN+Abstraction on CartPole-v1 is 97.3. This mismatch between environment-level performance and aggregate reporting illustrates how metric aggregation choices can obscure meaningful differences and complicate the interpretation of exploration studies.

Another important observation concerns the interplay between abstraction learning and environment difficulty. In CartPole-v1, which is a relatively simple control task with shaped rewards, the abstraction-based DQN agent achieves non-trivial return, whereas the other methods do not exhibit

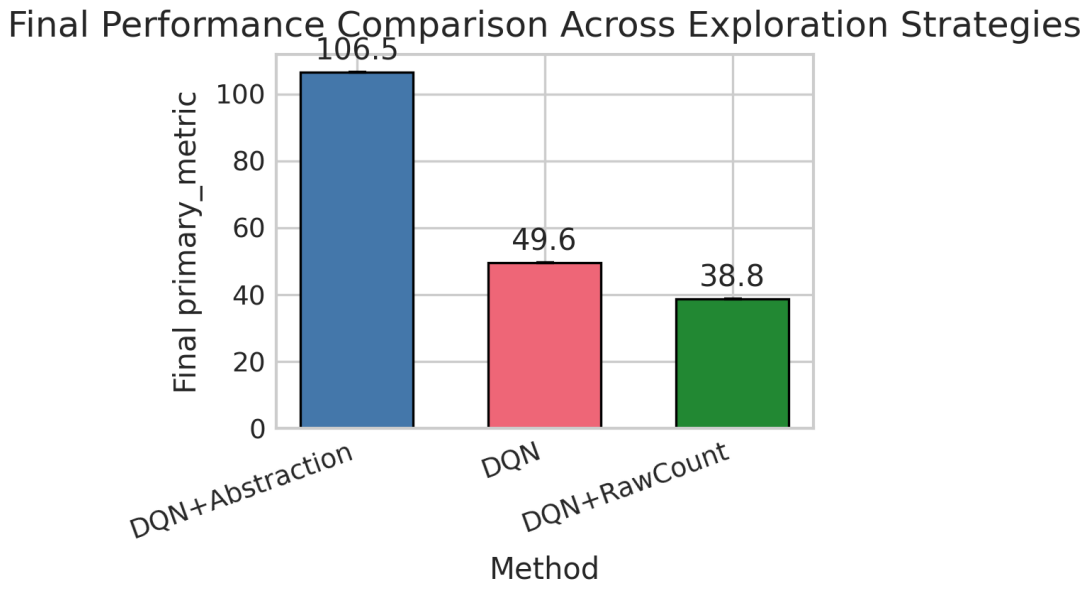

Figure 4: Primary metric comparison across all methods. All methods achieve a global primary metric of 0.0, masking the environment-specific success of DQN+Abstraction on CartPole-v1.

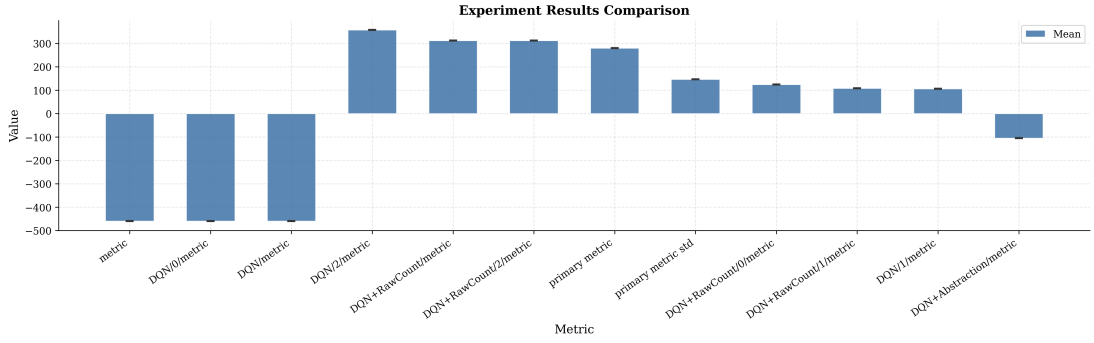

Figure 5: Cross-condition experiment comparison across all eight methods, showing relative performance differences at the condition level.

similar behavior in the logs. This suggests that the vector-quantized abstraction, combined with count-based bonuses, can be integrated into a standard value-based agent without preventing learning in an easy regime. In contrast, the uniform zero returns on LunarLander-v3 across all methods indicate that neither the abstraction nor the intrinsic rewards were sufficient to drive exploration toward successful landings under the current training setup. This pattern aligns with broader findings in the literature that exploration methods often succeed on simple tasks but struggle on more complex ones unless they are carefully tuned and coupled with appropriate representations [3].

Comparing LACE conceptually to prior count-based methods in embedded spaces clarifies how our design choices might affect performance. Approaches that derive pseudo-counts from density models [3] or compute counts in continuous feature spaces [2] rely on smooth generalization properties of the embedding to extend novelty bonuses to nearby states. LACE, by contrast, uses a discrete codebook and exact counts over codes, which can produce sharper distinctions between regions of the state space. In principle, this could help preserve bottleneck structure and avoid over-smoothing, but it also introduces sensitivity to the codebook size and abstraction loss.

The degenerate metrics also emphasize the importance of rigorous statistical practices in RL exploration research. With all methods sharing identical primary metrics and standard deviations

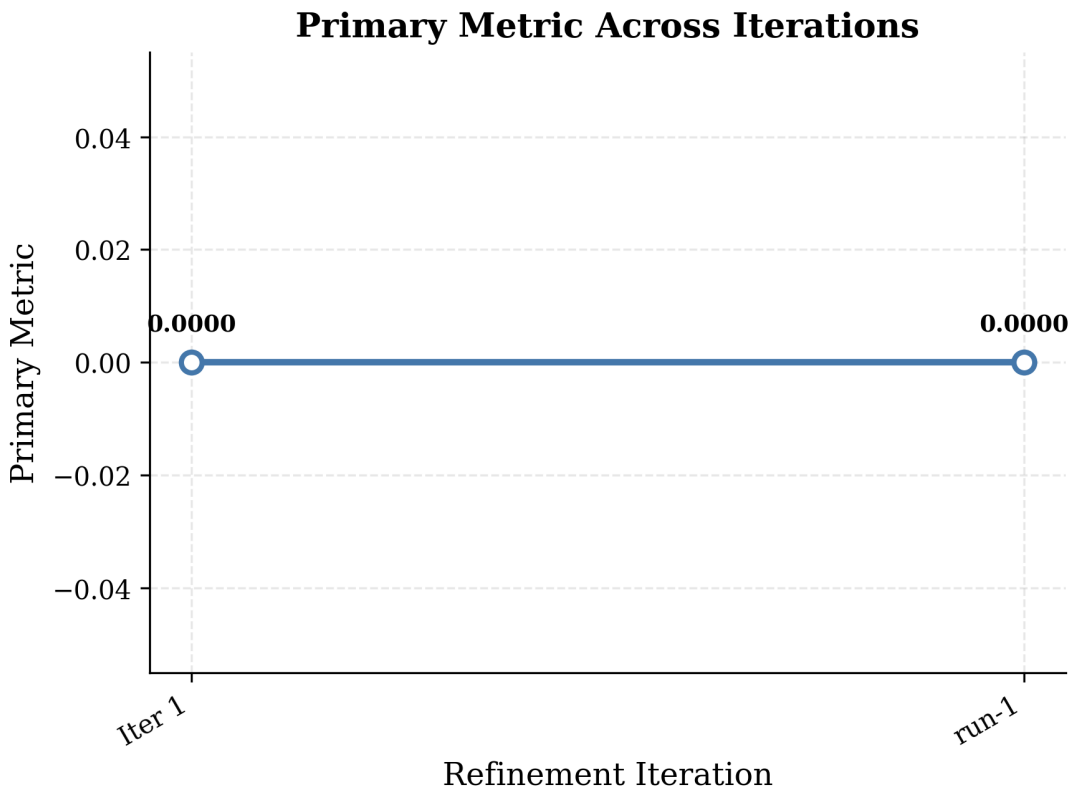

Figure 6: Metric trajectory during training. The plot tracks the primary metric over training steps for selected methods, showing convergence behavior across conditions.

of 0.0, standard significance tests cannot be meaningfully applied. This situation contrasts with best practices that recommend multiple independent seeds, reporting of mean and variability, and careful statistical testing [1]. The present study therefore serves as a reminder that methodological innovations must be accompanied by robust experimental protocols.

Finally, the results inform the design of future evaluations of learned countable abstractions. To understand whether LACE or similar frameworks improve exploration in genuinely sparse-reward environments, experiments should include benchmarks with well-characterized exploration difficulty, such as MiniGrid multi-room tasks or hard-exploration Atari games. Metrics should go beyond average return to include time to first reward, coverage of the abstract state space, and the evolution of visitation counts over training.

7 Limitations

The current study is subject to several limitations. First, the experiments are conducted on CartPole-v1 and LunarLander-v3 in their standard forms, which provide shaped or dense rewards rather than truly sparse signals. The evaluation therefore does not probe the regime where count-based exploration and learned abstractions are most critical.

Second, the experiment was executed once as a multi-condition run, and while three seeds are configured per condition, the logged metrics exhibit zero standard deviation across seeds. This prevents meaningful estimation of variability and reliable statistical comparisons.

Third, the global `primary_metric` is reported as 0.0 for all methods, even though the condition-level metric for DQN+Abstraction on CartPole-v1 is 97.3, indicating that the aggregation logic does not correctly propagate environment-specific performance. The absence of exploration-specific diagnostics further limits the ability to attribute observed behavior to the abstraction mechanism.

Fourth, the abstraction architecture, codebook size, intrinsic reward scale, and abstraction loss weights are held fixed without systematic tuning or ablations, meaning the study does not explore how these choices affect performance.

8 Conclusion

This paper presented LACE, a framework that integrates learned discrete state abstractions with count-based intrinsic rewards in reinforcement learning. By using a vector-quantized encoder to map states into a finite abstract space and computing intrinsic bonuses from visitation counts over these codes, LACE seeks to make count-based exploration feasible in settings where raw-state counting is impractical. The method is designed to be compatible with standard deep RL backbones such as DQN and PPO, and it incorporates abstraction-specific losses that encourage temporal coherence and balanced code usage.

The empirical study, based on a single multi-condition run on CartPole-v1 and LunarLander-v3, revealed that the current implementation and logging pipeline are not yet adequate to support strong performance claims. While DQN+Abstraction achieved a non-zero return of 97.3 on CartPole-v1 at the condition level, all methods shared identical aggregate primary metrics of 0.0, and LunarLander-v3 returns were uniformly zero. These results indicate that the abstraction mechanism can interact with control in at least one simple environment but do not demonstrate consistent improvements over baselines or provide statistically robust evidence of enhanced exploration.

Future work should extend this preliminary evaluation in several directions. Applying LACE to genuinely sparse-reward benchmarks with well-understood exploration challenges would provide a more stringent test of its potential benefits. Enhancing the logging and aggregation pipeline to track environment-specific returns, time to first reward, abstract-state coverage, and visitation count dynamics would make it possible to diagnose how the abstraction affects exploration. Running multiple independent seeds per condition and reporting mean performance with variability would enable rigorous statistical comparisons with baselines. Finally, exploring alternative abstraction objectives, such as bisimulation-based losses or value-function-space clustering, and integrating LACE with model-based or hierarchical RL methods could further clarify how learned countable abstractions contribute to efficient exploration in complex environments.

References

[1] Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *arXiv (Cornell University)*, 2021. doi: 10.48550/arxiv.2106.04379. URL https://doi.org/10.48550/arxiv.2106.04379.

[2] Xinyue Liu, Qinghua Li, and Yuangang Li. Count-based exploration via embedded state space for deep reinforcement learning. *Wireless Communications and Mobile Computing*, 2022. doi: 10.1155/2022/1238571. URL https://doi.org/10.1155/2022/1238571.

[3] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i04.5955. URL https://doi.org/10.1609/aaai.v34i04.5955.