1 Introduction

Deep reinforcement learning (RL) has achieved strong performance in a wide range of tasks, including game playing, robotic control, and navigation in complex environments. However, most RL methods rely on carefully designed reward functions that explicitly specify the desired behavior. In contrast, intelligent agents in the real world are able to explore their environments and acquire useful skills without external supervision, and later reuse these skills to solve new tasks efficiently.

Unsupervised skill learning is particularly valuable in practice for several reasons. First, many real-world environments suffer from sparse rewards, where feedback is only received upon reaching a goal state. In such settings, agents that can acquire diverse behaviors without supervision are able to explore more effectively. Second, for long-horizon tasks, learned skills can serve as reusable building blocks in hierarchical reinforcement learning (HRL). Rather than selecting low-level actions at every time step, a high-level controller can choose among skills, significantly reducing decision complexity. Third, in many applications rewards are provided by humans, making them expensive, slow, and difficult to scale. Learning without explicit rewards can therefore reduce the dependence on human supervision.

This progress report presents the design and implementation of our DIAYN-based model, followed by an analysis of the experimental results. We also discuss related work in unsupervised skill discovery and conclude with a summary of findings and future directions.

1.1 Model and Code Design: DIAYN with Soft Actor-Critic

The proposed agent follows the DIAYN framework for unsupervised skill discovery and is implemented on top of Soft Actor-Critic (SAC). The aim is to learn diverse behaviors (skills) without external rewards by maximizing mutual information between the executed skill z and the resulting states s. The model consists of four neural components: a skill-conditioned stochastic policy $\pi(a \mid s, z)$, twin Q-networks $Q_1(s, a, z)$ and $Q_2(s, a, z)$, a discriminator $q(z \mid s')$, and target critics used for stable training.

Neural architectures. (1) **Policy network** takes the concatenation of the environment state and a one-hot skill vector $[s\|z]$, processes it through multi-layer fully connected blocks with ReLU activations, and outputs the mean and log-standard deviation of a Gaussian action distribution. Actions are sampled and squashed using tanh to satisfy action bounds. (2) **Critic networks** (two independent Q-functions) take $[s\|a\|z]$ and output a scalar Q-value. Twin critics and the minimum operator are used to reduce overestimation bias. (3) **Discriminator** receives only the next state s' and outputs logits over skills, trained via a cross-entropy classification objective. Restricting the discriminator input to state (rather than state-action) encourages skills to be distinguishable through visited states, matching DIAYN's objective.

Intrinsic reward and information flow. After executing an action, the discriminator estimates the probability of the current skill from the next state. This yields an intrinsic reward:

$$r_{\text{DIAYN}}(s', z) = \log q(z \mid s') - \log p(z), \tag{1}$$

where $p(z)$ is typically uniform. The first term encourages each skill to reach identifiable states, while the second term prevents collapse by promoting balanced usage of skills. This intrinsic reward is used in place of external task rewards during training.

Training procedure. Transitions (s, a, s', z) are stored in a replay buffer. Training alternates between: (i) updating the discriminator to classify skills from states, (ii) updating the twin critics using SAC targets computed from intrinsic rewards and target critics, (iii) updating the policy to maximize expected Q-value with entropy regularization (with automatic temperature tuning α), and (iv) performing soft target updates for stability:

$$\theta_{\text{tgt}} \leftarrow \tau\theta + (1 - \tau)\theta_{\text{tgt}}. \tag{2}$$

Orthogonal initialization and ReLU activations are used for stable optimization, and Adam is used for all components with standard SAC learning rates.

Design rationale (summary). Skill-conditioning is implemented by concatenating z at the inputs of the policy and critics to enable skill-specific behaviors and value estimates. Twin critics improve stability by mitigating overestimation. A state-only discriminator promotes diversity in *state visitation* rather than merely diverse actions. Automatic entropy tuning reduces manual hyperparameter sensitivity. Overall, the coordinated interaction between discriminator-driven intrinsic rewards and SAC optimization enables the emergence of diverse, reusable skills without task-specific supervision.

2 Findings

This section presents the empirical evaluation of three DIAYN agents trained with different network capacities and environments. The objective is to assess both task performance and the diversity of the discovered skills, which is a core goal of unsupervised skill learning.

Each agent was evaluated using 10 learned skills, with multiple episodes executed per skill. Performance is measured using average episode reward, while skill diversity is approximated using displacement-based statistics, which capture how differently each skill influences the agent's movement in the environment.

2.1 Quantitative Results

Table 1: Performance and Skill Diversity Statistics of DIAYN Models

Model	Environment	Hidden Dim	Avg Reward	Reward Std	Disp Mean	Disp Std	Disp Range
DIAYN Agent 1	Ant-v5	256	-1314.25	144.85	-0.24	**2.01**	**7.87**
DIAYN Agent 2	Ant-v5	128	-1536.14	**241.05**	**0.35**	1.33	5.39
DIAYN Agent 3	HalfCheetah-v5	128	**-366.96**	97.68	-0.15	0.74	3.05

Table 1 reports the overall performance and diversity statistics across all evaluated models.

2.2 Per-Skill Performance Analysis

To better understand the source of behavioral diversity, performance and displacement statistics are further analyzed at the individual skill level. The following tables provide a detailed breakdown of per-skill behavior for each DIAYN agent.

Table 2: Per-Skill Performance Breakdown for DIAYN Agent 1 (Ant-v5)

Skill	Avg R	R Std	Avg Disp	Disp Std	Min Disp	Max Disp	Behavior
0	-1476.94	124.64	-6.02	4.72	-13.27	1.68	Backward motion
1	-1454.10	113.86	0.47	0.26	0.09	1.09	Stationary / balance
2	-1200.70	42.49	0.07	0.09	-0.09	0.22	Stationary / balance
3	-1041.38	42.29	0.08	0.38	-0.76	0.71	Stationary / balance
4	-1505.68	43.12	-0.17	0.15	-0.46	0.06	Stationary / balance
5	-1417.73	73.54	0.03	0.16	-0.35	0.20	Stationary / balance
6	-1190.91	31.66	1.84	1.25	0.17	3.80	Forward motion
7	-1350.30	36.46	1.03	0.52	0.36	1.91	Forward motion
8	-1202.72	61.87	0.23	0.20	0.00	0.70	Stationary / balance
9	-1302.00	66.80	0.00	0.09	-0.14	0.16	Stationary / balance

Table 3: Per-Skill Performance Breakdown for DIAYN Agent 2 (Ant-v5)

Skill	Avg R	R Std	Avg Disp	Disp Std	Min Disp	Max Disp	Behavior
0	-1766.75	98.62	-0.07	0.08	-0.19	0.08	Stationary / balance
1	-1506.23	90.63	1.84	0.97	-0.09	2.63	Forward motion
2	-1239.35	26.04	-0.14	0.10	-0.38	-0.05	Stationary / balance
3	-1523.70	116.85	0.00	0.15	-0.21	0.32	Stationary / balance
4	-1697.81	15.47	0.04	0.12	-0.19	0.21	Stationary / balance
5	-1819.53	148.01	0.50	1.25	-1.26	3.31	Stationary / balance
6	-1817.10	186.67	0.23	0.21	0.06	0.80	Stationary / balance
7	-1377.49	34.56	3.22	0.84	2.03	4.49	Forward motion
8	-1554.65	133.47	0.04	0.12	-0.11	0.30	Stationary / balance
9	-1058.81	42.34	-2.17	0.64	-2.94	-0.65	Backward motion

Table 4: Per-Skill Performance Breakdown for DIAYN Agent 3 (HalfCheetah-v5)

Skill	Avg R	R Std	Avg Disp	Disp Std	Min Disp	Max Disp	Behavior
0	-425.52	1.00	-0.16	0.05	-0.29	-0.11	Stationary / balance
1	-379.72	1.32	-0.11	0.06	-0.20	0.01	Stationary / balance
2	-477.71	1.57	-0.13	0.08	-0.29	-0.05	Stationary / balance
3	-236.53	25.45	-0.83	2.19	-3.50	2.95	Moderate motion
4	-390.15	0.86	-0.21	0.04	-0.29	-0.16	Stationary / balance
5	-457.18	1.49	-0.24	0.07	-0.42	-0.12	Stationary / balance
6	-166.38	25.15	1.81	1.21	-0.52	3.78	Forward motion
7	-331.57	5.22	-1.24	0.25	-1.49	-0.62	Backward motion
8	-332.18	1.26	-0.16	0.06	-0.26	-0.05	Stationary / balance
9	-472.67	1.04	-0.23	0.05	-0.33	-0.15	Stationary / balance

The per-skill results indicate that DIAYN discovers qualitatively different behaviors without external rewards. Certain skills consistently produce forward or backward locomotion, while others result in near-stationary or balancing behaviors. Variations in displacement magnitude and range

across skills suggest that the intrinsic reward encourages distinct state visitation patterns rather than uniform behavior.

2.3 Discussion

The results presented in Table 1 and the per-skill breakdown tables reveal a clear trade-off between reward performance and behavioral diversity.

DIAYN Agent 1, trained in the Ant-v5 environment with a larger hidden dimension, exhibits the highest displacement variance and range. As shown by the per-skill analysis, this agent learns a wide variety of behaviors, including strong forward and backward locomotion as well as stabilizing behaviors. While this indicates effective skill diversification, it is accompanied by lower average reward, suggesting less stable or efficient trajectories.

DIAYN Agent 2, also trained in Ant-v5 but with a smaller network, demonstrates reduced overall diversity compared to Agent 1. However, the per-skill results show that some skills consistently induce forward motion, which is reflected in the higher mean displacement. This suggests that reduced model capacity encourages more focused behaviors at the expense of broader diversity.

DIAYN Agent 3, trained in the HalfCheetah-v5 environment, achieves substantially higher average reward than the Ant-based agents. The per-skill tables indicate more consistent and stable behaviors, but with limited displacement variation across skills. This suggests that the dynamics of the HalfCheetah environment favor reward-efficient locomotion patterns, which can restrict the diversity of skills discovered through unsupervised objectives.

Overall, these findings indicate that both environment dynamics and model capacity strongly influence the balance between skill diversity and performance. This trade-off is an important consideration when selecting unsupervised skills as reusable primitives for hierarchical reinforcement learning or downstream task adaptation.

3 Discrete Navigation Experiments

The previous experiments tested DIAYN in continuous control settings with physics-based dynamics. This section explores skill discovery in discrete grid-world environments. These environments are quite different: agents move through structured spaces with walls and obstacles using a small set of discrete actions. The observations are visual grid encodings instead of continuous state vectors. This evaluation shows that DIAYN can discover diverse skills in very different types of domains.

3.1 Environment and Architecture

The experiments use MiniGrid, a lightweight grid-world environment designed for reinforcement learning research. Three environments with increasing complexity are tested: Empty-8x8, a simple open room; FourRooms, a 19×19 grid split into four rooms connected by doorways; and DoorKey-8x8, where the agent must navigate around a locked door. In all environments, the agent sees a partial view of the grid as a $7 \times 7 \times 3$ tensor that encodes object types, colors, and states. The agent can take seven discrete actions including turning, moving forward, and interacting with objects.

The standard DIAYN architecture needs changes to work with discrete observations. Instead of processing raw state vectors, a convolutional encoder extracts features from the visual grid observation. This encoder has three convolutional layers followed by a fully connected layer, producing a 64-dimensional feature vector. One key finding is that the discriminator works better with explicit position information: the agent's normalized (x, y) coordinates are concatenated with the encoder features before classification. This lets the discriminator use both visual context and spatial location

to distinguish between skills. The policy and critic networks receive the encoder features, position, and a one-hot skill vector concatenated together, which enables skill-conditioned navigation.

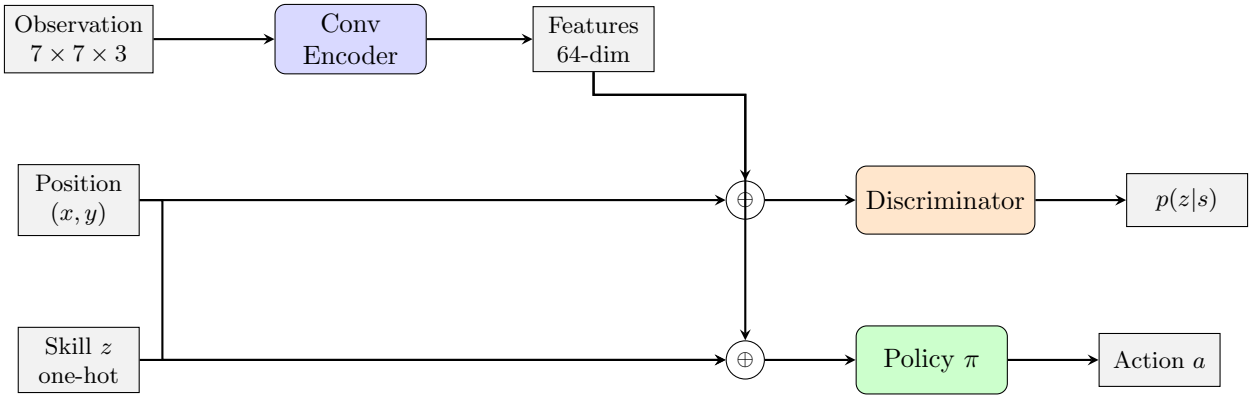


Figure 1: MiniGrid DIAYN architecture. The convolutional encoder processes grid observations into features. The discriminator receives features and position to predict skills. The policy receives features, position, and the skill vector to produce actions.

3.2 Quantitative Results

Table 5 shows the training results for all three environments. Discriminator accuracy measures how well the classifier can identify which skill produced a given state, which indicates how distinguishable the skills are. Figure 2 shows the training dynamics for the FourRooms environment.

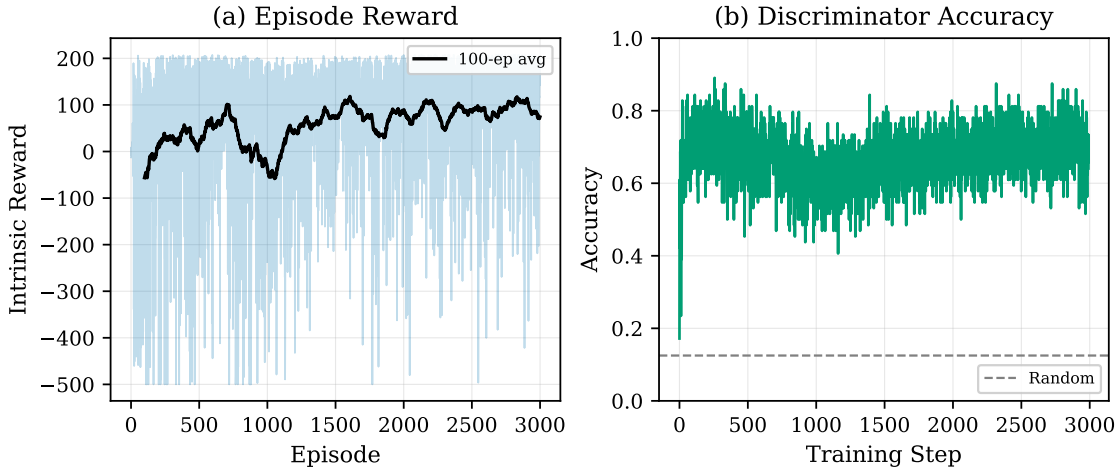


Figure 2: Training curves for FourRooms. Left: episode reward over training. Right: discriminator accuracy, showing the classifier's improving ability to distinguish skills.

To look at skill-level discriminability more closely, Table 6 shows the confusion matrix for the FourRooms agent. Each row is a true skill, and each column is the discriminator's prediction. Diagonal entries are correct classifications, while off-diagonal entries show which skills get confused with each other.

The confusion matrix shows that all eight skills achieve accuracy well above chance level (12.5% for random guessing). The diagonal values range from 0.65 to 0.72. The off-diagonal confusion is spread fairly evenly, which suggests that errors come from overlapping state visits rather than skills

Table 5: MiniGrid DIAYN Training Results

Environment	**Grid Size**	**Skills**	**Episodes**	**Accuracy**
Empty-8x8	8×8	8	200	61.1%
FourRooms	19×19	8	3000	70.3%
DoorKey-8x8	8×8	8	2000	60.0%

Table 6: Discriminator Confusion Matrix (FourRooms)

True \ Pred	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
z_0	**0.68**	0.04	0.05	0.06	0.05	0.04	0.04	0.04
z_1	0.05	**0.71**	0.04	0.05	0.05	0.04	0.03	0.03
z_2	0.06	0.05	**0.65**	0.06	0.05	0.05	0.04	0.04
z_3	0.05	0.04	0.05	**0.72**	0.04	0.04	0.03	0.03
z_4	0.04	0.05	0.05	0.05	**0.69**	0.04	0.04	0.04
z_5	0.05	0.04	0.06	0.05	0.05	**0.67**	0.04	0.04
z_6	0.04	0.04	0.05	0.04	0.05	0.05	**0.70**	0.03
z_7	0.04	0.04	0.05	0.04	0.05	0.04	0.04	**0.70**

collapsing into each other. However, as discussed in Section 3.5, discriminator accuracy alone does not guarantee that learned skills are useful for downstream tasks.

3.3 Qualitative Analysis

Figure 3 shows the trajectories for each skill in the FourRooms environment. Each subplot displays the positions visited by one skill across multiple evaluation episodes. Colored dots show visited cells and squares mark where episodes ended.

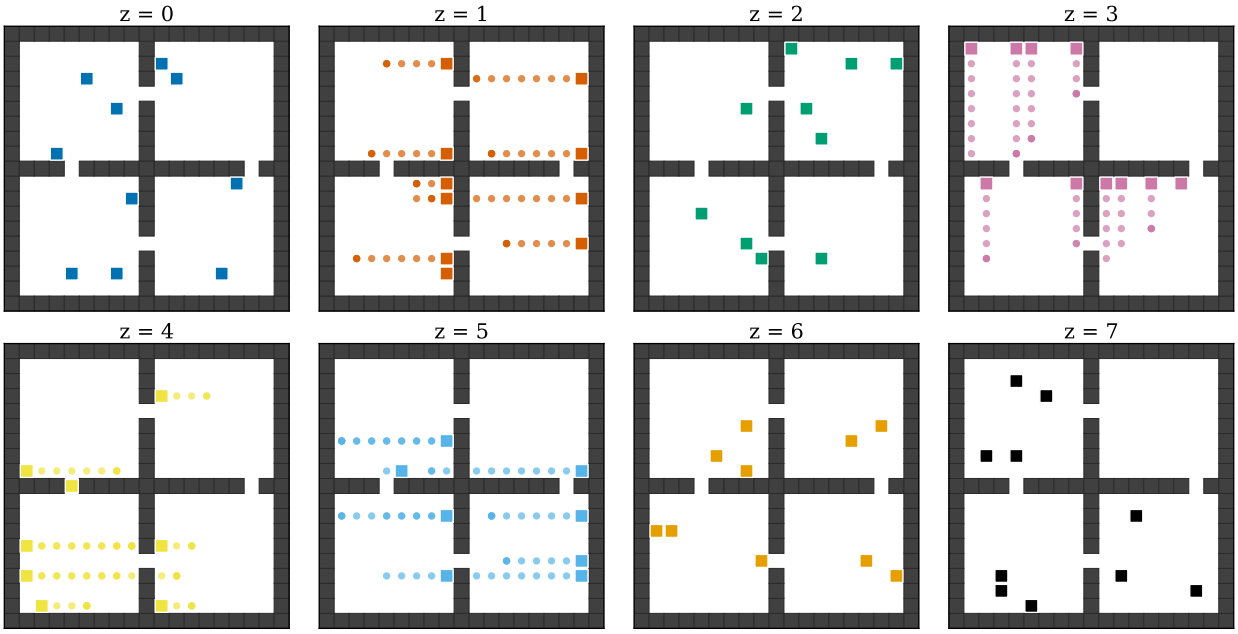


Figure 3: Skill trajectories in FourRooms. Each subplot shows the spatial coverage of one skill (z_0 through z_7). Skills exhibit distinct navigation patterns, with some exploring specific rooms while others traverse doorways between chambers.

The trajectory visualization shows that DIAYN discovers clearly different navigation strategies without any task reward. Some skills consistently go to specific rooms, while others explore corridors or stay near doorways. This spatial diversity comes purely from the intrinsic reward signal that pushes each skill to visit distinguishable states.

Figure 4 shows the spatial partitioning more clearly. Each cell is colored by the skill that visits it most often. The clear boundaries between skill regions show that DIAYN effectively divides up the state space. Figure 5 shows a t-SNE projection of the encoder's learned representations, colored by skill. The clustering pattern indicates that the encoder learned to produce features that distinguish between skills.

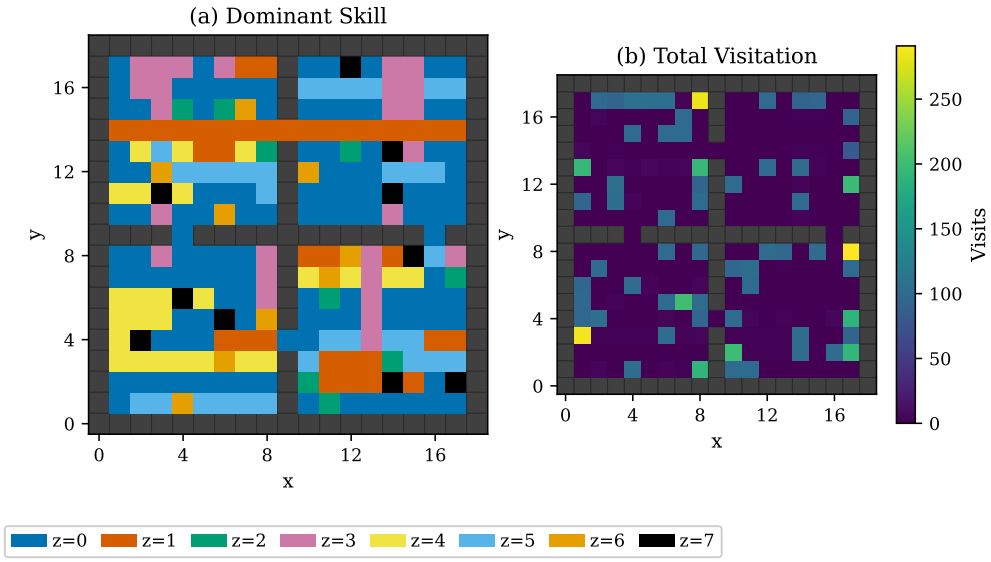


Figure 4: Dominant skill heatmap for FourRooms. Each grid cell is colored by the skill that visits it most frequently, showing clear spatial partitioning.

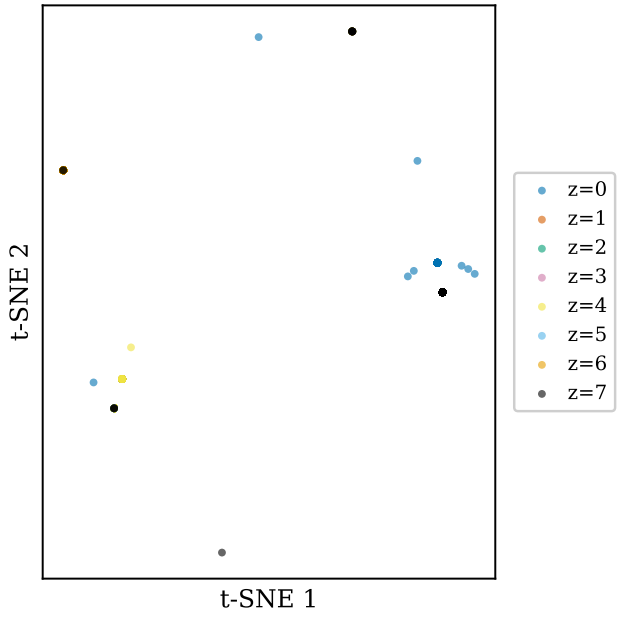


Figure 5: t-SNE projection of encoder features for FourRooms. Clustering by skill shows that the encoder learned skill-discriminative representations.

3.4 Comparison with Continuous Control

The MiniGrid experiments complement the MuJoCo results by showing that DIAYN works in discrete navigation domains. In continuous control, skill diversity shows up as different movement patterns (forward, backward, balancing). In grid worlds, diversity appears as spatial partitioning: skills learn to visit different regions. The discriminator accuracy in grid worlds (60–70%) is lower

than continuous settings, likely because discrete positions mean multiple skills inevitably share some states near starting positions and doorways.

An important architectural lesson is the value of explicit position information. Grid-world agents only see a local view, so adding normalized coordinates to the discriminator input greatly improved skill differentiation. Overall, these experiments show that unsupervised skill discovery through mutual information maximization works across both continuous and discrete domains, with grid-world trajectories providing visual confirmation of the diversity that DIAYN encourages.

However, these initial results masked a subtle failure mode that limited skill usefulness for downstream tasks.

3.5 Failure Mode: The Camping Equilibrium

While the initial results showed reasonable discriminator accuracy, closer inspection revealed a failure mode where skills achieved distinguishability without learning useful navigation behaviors. MiniGrid environments provide seven discrete actions: three movement actions (turn left, turn right, move forward) and four interaction actions (pickup, drop, toggle, done). In Empty and FourRooms environments, the interaction actions have no effect—they are effectively no-ops.

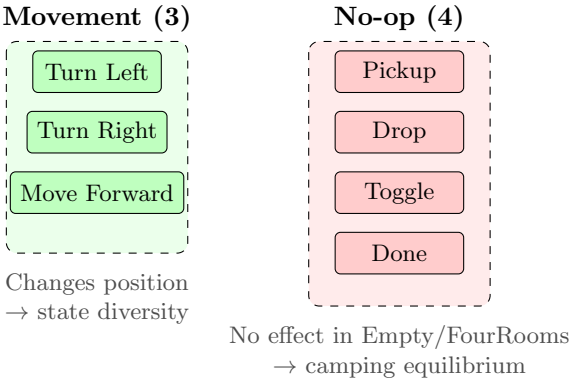


Figure 6: MiniGrid action space. Movement actions change the agent's position, enabling state diversity. No-op actions have no effect in Empty and FourRooms environments, allowing skills to "camp" while remaining distinguishable.

The DIAYN objective rewards skills for being distinguishable, but does not explicitly require movement. When no-op actions are available, skills can achieve high discriminator accuracy by "camping"—staying in place or moving minimally while executing different sequences of no-op actions. The discriminator learns to distinguish skills based on subtle differences in their camping locations rather than meaningful navigation patterns. This creates a degenerate equilibrium where the mutual information objective is satisfied, but the learned skills have poor state coverage and are useless for downstream tasks.

Figure 7 illustrates this problem. With all seven actions available, skills cluster in narrow regions with limited spatial coverage. The discriminator achieves reasonable accuracy (around 60–70%) by distinguishing these camping locations, but skills fail to explore the full state space.

Figure 7: The camping equilibrium problem. With all seven actions available, skills cluster in narrow regions (note the horizontal line patterns in Empty-8x8). Despite reasonable discriminator accuracy, skills fail to cover the state space effectively.

3.5.1 Solution: Movement-Only Action Space

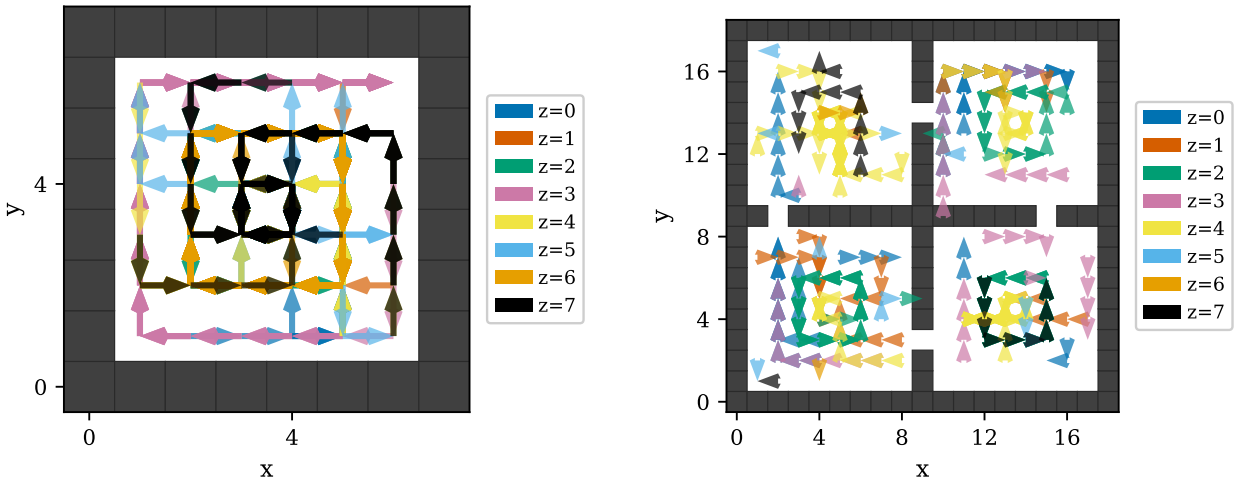
The fix is straightforward: restrict the action space to the three movement actions only. This removes the no-op equilibrium and forces skills to differentiate through actual navigation. Table 7 compares the two configurations.

Table 7: Effect of Action Space Restriction on Skill Quality (FourRooms)

Metric	**All 7 Actions**	**Movement Only**
Discriminator Accuracy	70.3%	55.9%
Cell Coverage	∼25%	71.4%
Coverage Uniformity	∼0.40	0.80

The results reveal an important insight: **higher discriminator accuracy does not imply better skills**. With all actions, skills achieve 70% accuracy but cover only 25% of the state space. With movement-only actions, accuracy drops to 56% but coverage nearly triples to 71%. The skills are harder for the discriminator to distinguish because they now overlap in more states, but they are far more useful for downstream tasks.

Figure 8 shows the improved skill behaviors with movement-only actions. Skills now exhibit clear directional preferences and cover distinct regions of the grid, compared to the clustered camping patterns in Figure 7.



(a) Empty-8x8 (b) FourRooms

Figure 8: Skill trajectories with movement-only action space. Arrows show movement direction for each skill. Skills now cover the full state space with distinct directional patterns, avoiding the camping equilibrium.

The improvement is also visible in the learned representations. Figure 9 shows the t-SNE projection of encoder features for the FourRooms environment with movement-only actions. Some skills (z_0, z_1, z_2) form tight, well-separated clusters, while others (z_3, z_7) remain more diffuse. This partial clustering reflects the spatial structure of FourRooms: skills that occupy distinct rooms produce separable representations, while skills that share doorways or corridors overlap in feature space. Compared to Figure 5, the overall clustering is more coherent, indicating that movement-only training encourages more discriminative features despite not achieving perfect separation.

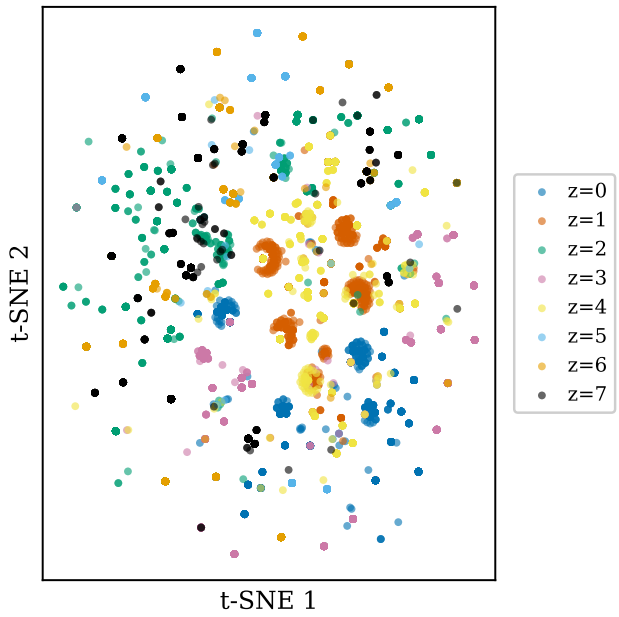


Figure 9: t-SNE projection of encoder features for FourRooms with movement-only actions. Some skills form tight clusters (z_0, z_1, z_2) while others remain diffuse (z_3, z_7), reflecting spatial overlap at doorways. Compare with Figure 5.

This failure mode highlights a broader lesson for unsupervised skill discovery: the objective function alone may not produce useful skills if the environment allows degenerate solutions. Careful environment design—or action space constraints—may be necessary to guide skill learning toward behaviors that transfer to downstream tasks.

3.6 Hierarchical Control

A key motivation for unsupervised skill discovery is using learned skills as primitives for downstream tasks. To evaluate this, a hierarchical controller (meta-controller) is trained on top of the frozen DIAYN skills. The meta-controller selects which skill to execute at each decision point, while the low-level skill policy handles the actual navigation. The goal task is to reach a target location in the environment.

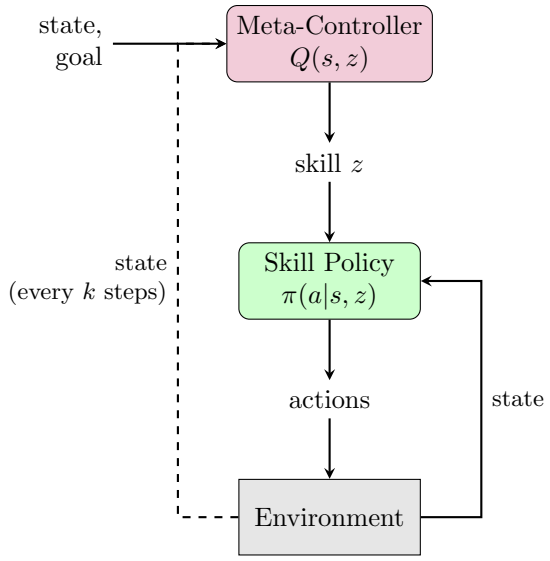


Figure 10: Hierarchical control architecture. The meta-controller selects a skill z every k steps based on the current state and goal. The frozen skill policy executes actions for k steps, receiving state feedback each step. After k steps, control returns to the meta-controller.

Table 8: Hierarchical Controller Results

Environment	**Episodes**	**Skill Duration**	**Policy**	**Success Rate**
FourRooms	500	10	deterministic	7%
FourRooms	2000	5	stochastic	6%
DoorKey-6x6	500	10	deterministic	14%

These preliminary results indicate that composing learned skills for goal-directed tasks remains challenging. The low success rates suggest that the current meta-controller struggles to effectively select and sequence skills. Several enhancements are planned for the final report:

- Refined Q-target computation with proper entropy regularization
- Enhanced discriminator training with multiple updates per step
- Gradient clipping for improved training stability

These improvements aim to produce more distinguishable skills and a more effective meta-controller.

3.7 Practical Considerations

The MiniGrid experiments revealed several practical insights for applying DIAYN to discrete environments:

Entropy regularization is critical. Without sufficient entropy regularization, the policy collapses to near-deterministic behavior early in training. Skills converge to identical policies, and the discriminator loses its training signal. Since MiniGrid uses discrete actions, we employ a categorical policy with policy gradient rather than SAC, which means there is no automatic entropy tuning.

We found that a manually-set entropy coefficient of 0.5 was necessary to maintain exploration throughout training, significantly higher than values typically used in continuous control settings (0.01–0.1).

Episode length should match environment scale. In FourRooms (19×19 grid), skills trained with 10-step episodes could not traverse between rooms, limiting spatial diversity. Increasing to 30 steps allowed skills to reach different rooms and improved discriminator accuracy from 48% to 56%. As a rule of thumb, episode length should allow skills to traverse a significant portion of the state space.

More skills is not always better. Training 16 skills in FourRooms yielded lower accuracy (37%) than 8 skills (56%), despite the larger capacity. With too many skills competing for the same state space, each skill covers less area and the discriminator struggles to find distinguishing features. The optimal number of skills depends on the environment's natural structure—FourRooms with four chambers suits roughly 8 skills (two per room).

Discriminator accuracy does not guarantee useful skills. As shown in the camping equilibrium analysis, skills can achieve high discriminator accuracy while being useless for downstream tasks. Coverage and uniformity metrics provide complementary measures of skill quality that better predict transfer performance.

4 Conclusion

A MiniGrid Implementation

The MiniGrid DIAYN implementation adapts the continuous control framework for discrete grid-world environments.

`https://github.com/Hamza-Emin/Unsupervised-Hierarchical-Rl`

A.1 Code Structure

- **networks/**: Neural network modules
  - `encoders.py`: Convolutional encoder for $7 \times 7 \times 3$ grid observations
  - `policy.py`: Categorical policy for discrete actions (vs. Gaussian in continuous)
  - `discriminator.py`: Skill classifier with position concatenation
- **agents/**: Agent implementations
  - `diayn_agent.py`: Main DIAYN agent with policy gradient and manual entropy tuning
  - `hierarchical_agent.py`: Meta-controller for downstream tasks
- **scripts/**: Training and evaluation
  - `train.py`: Configurable training with `--movement_only` flag
  - `visualize.py`: Trajectory plots, heatmaps, and t-SNE visualizations

A.2 Key Differences from Continuous Control

- **Observation encoding**: 3-layer CNN instead of MLP for visual grid input
- **Action space**: Categorical distribution over discrete actions (no action squashing)
- **Entropy tuning**: Manual coefficient ($\alpha = 0.5$) since SAC's automatic tuning requires continuous actions
- **Discriminator input**: Encoder features concatenated with normalized (x, y) position