Contrastive Intrinsic Control for Unsupervised Reinforcement Learning

Michael Laskin
UC Berkeley
mlaskin@berkeley.edu

Hao Liu
UC Berkeley

Xue Bin Peng
UC Berkeley

Denis Yarats
NYU, Meta AI

Aravind Rajeswaran
UC Berkeley, Meta AI

Pieter Abbeel
UC Berkeley, Covariant

Abstract

We introduce Contrastive Intrinsic Control (CIC), an unsupervised reinforcement learning (RL) algorithm that maximizes the mutual information between state-transitions and latent skill vectors. CIC utilizes contrastive learning between state-transitions and skills vectors to learn behaviour embeddings and maximizes the entropy of these embeddings as an intrinsic reward to encourage behavioural diversity. We evaluate our algorithm on the Unsupervised RL Benchmark (URLB) in the asymptotic state-based setting, which consists of a long reward-free pre-training phase followed by a short adaptation phase to downstream tasks with extrinsic rewards. We find that CIC improves over prior exploration algorithms in terms of adaptation efficiency to downstream tasks on state-based URLB. [1]

Deep RL is a powerful approach toward solving complex control tasks in the presence of extrinsic rewards. Successful applications include playing video games from pixels [1], mastering the game of Go [2, 3], robotic locomotion [4, 5, 6] and dexterous manipulation [7, 8, 9] policies. While effective, the above advances produced agents that are unable to generalize to new downstream tasks beyond the one they were trained to solve. Humans and animals on the other hand are able to acquire skills with minimal supervision and apply them to solve a variety of downstream tasks. In this work, we seek to train agents that acquire skills without supervision with generalization capabilities by efficiently adapting these skills to downstream tasks.

Over the last few years, unsupervised RL has emerged as a promising framework for developing RL agents that can generalize to new tasks. In the unsupervised RL setting, agents are first pre-trained with self-supervised intrinsic rewards and then finetuned to downstream tasks with extrinsic rewards. Unsupervised RL algorithms broadly fall into three categories - knowledge-based, data-based, and competence-based methods[2]. Knowledge-based methods maximize the error or uncertainty of a predictive model [12, 13, 14]. Data-based methods maximize the entropy of the agent's visitation [15, 16]. Competence-based methods learn skills that generate diverse behaviors [17, 18]. This work falls into the latter category of competence-based exploration methods.

Unlike knowledge-based and data-based algorithms, competence-based algorithms simultaneously address both the exploration challenge as well as distilling the generated experience in the form of reusable skills. This makes them particularly appealing, since the resulting skill-based policies (or skills themselves) can be finetuned to efficiently solve downstream tasks. While there are many self-supervised objectives that can be utilized, our work falls into a family of methods that learns skills by maximizing the mutual information between visited states and latent skill vectors. Many earlier

[1]Project website and code: https://sites.google.com/view/cicneurips2022/
[2]These categories for exploration algorithms were introduced by [10] and inspired by [11].

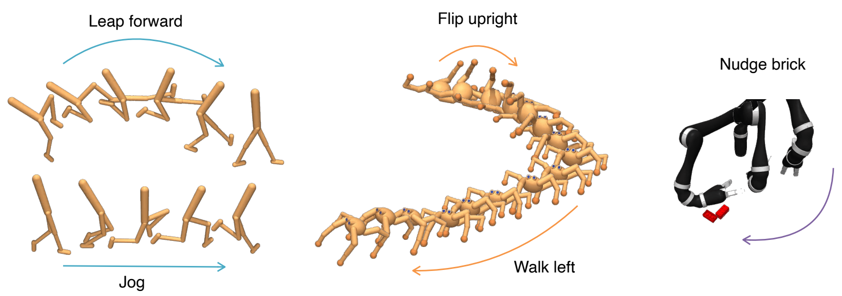

Figure 1: Qualitative visualizations of unsupervised skills discovered in Walker, Quadruped, and Jaco arm environments. The Walker learns to balance and move, the Quadruped learns to flip upright and walk, and the 6 DOF robotic arm learns how to move without locking. Unlike prior competence-based methods for continuous control which evaluate on OpenAI Gym (e.g. [17]), which reset the environment when the agent loses balance, CIC is able to learn skills in fixed episode length environments which are much harder to explore (see Appendix I).

works have investigated optimizing such objectives [17, 18, 19, 20]. However, competence-based methods have been empirically challenging to train and have under-performed when compared to knowledge and data-based methods [21].

In this work, we take a closer look at the challenges of pre-training agents with competence-based algorithms. We introduce Contrastive Intrinsic Control (CIC) – an exploration algorithm that uses a new estimator for the mutual information objective. CIC combines particle estimation for state entropy [22, 15] and noise contrastive estimation [23] for the conditional entropy which enables it to both generate diverse behaviors *(explore)* and discriminate high-dimensional continuous skills *(exploit)*. To the best of our knowledge, CIC is the first exploration algorithm to utilize noise contrastive estimation to discriminate between state transitions and latent skill vectors. Empirically, we show that CIC adapts to downstream tasks more efficiently than prior exploration approaches on the state-based Unsupervised Reinforcement Learning Benchmark (URLB). CIC achieves 79% higher returns on downstream tasks than prior competence-based algorithms and 18% higher returns than the next-best exploration algorithm overall.

1 Background and Notation

Markov Decision Process: We operate under the assumption that our system is described by a Markov Decision Process (MDP) [24]. An MDP consists of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ which has states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $p(s'|s,a) \sim \mathcal{P}$, a reward function r, and a discount factor γ. In an MDP, at each timestep t, an agent observes the current state s, selects an action from a policy $a \sim \pi(\cdot|s)$, and then observes the reward r and next state s' once it acts in the environment. Note that usually r refers to an extrinsic reward. However, in this work we will first be pre-training an agent with intrinsic rewards r^{int} and finetuning on extrinsic rewards r^{ext}.

For convenience we also introduce the variable $\tau = (s, s')$ which is a tuple denoting a transition between two consecutive states. Importantly, τ does not denote a state-action trajectory. In addition to the standard MDP notation, we will also be learning skills $z \in \mathcal{Z}$ where $\mathcal{Z}$ is the skill set, which can be a discrete or continuous real-valued vector space, and our policy will be skill-conditioned $a \sim \pi(\cdot|s, z)$.

Unsupervised Skill Discovery through Mutual Information Maximization: Most competence-based approaches to exploration maximize the mutual information between states and skills. Our work and a large body of prior research [17, 20, 18, 25, 26, 27] aims to maximize a mutual information objective with the following general form:

$$I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z) \tag{1}$$

Competence-based algorithms use different choices for τ and can condition on additional information such as actions or starting states. For a full summary of competence-based algorithms and their objectives see Table 2.

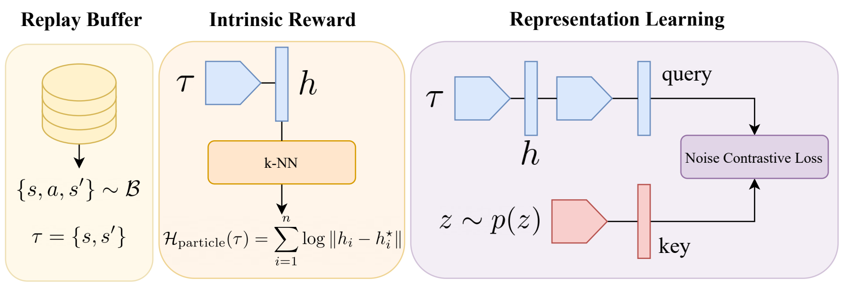

Figure 2: Architecture illustrating the practical implementation of CIC . During a gradient update step, random $\tau = (s, s')$ tuples are sampled from the replay buffer, then a particle estimator is used to compute the entropy and a noise contrastive loss to compute the conditional entropy. The contrastive loss is backpropagated through the entire architecture. The entropy and contrastive terms are then scaled and added to form the intrinsic reward. The RL agent is optimized with a DDPG [29].

Lower Bound Estimates of Mutual Information: The mutual information $I(s; z)$ is intractable to compute directly. Since we wish to maximize $I(s; z)$, we can approximate this objective by instead maximizing a lower bound estimate. Most known mutual information maximization algorithms use the variational lower bound introduced in [28]:

$$I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z) \geq \mathcal{H}(\tau) + \mathbb{E}_{\tau,z}[\log q(\tau|z)] \tag{2}$$

The variational lower bound can be applied to both decompositions of the mutual information. The design decisions of a competence-based algorithm therefore come down to (i) which decomposition of $I(\tau; z)$ to use, (ii) whether to use discrete or continuous skills, (iii) how to estimate $H(z)$ or $H(\tau)$, and finally (iv) how to estimate $H(z|\tau)$ or $H(\tau|z)$.

2 Motivation

Results from the recent Unsupervised Reinforcement Learning Benchmark (URLB) [21] show that competence-based approaches underperform relative to knowledge-based and data-based baselines on DeepMind Control (DMC). We argue that the underlying issue with current competence-based algorithms when deployed on harder exploration environments like DMC has to do with the currently used estimators for $I(\tau; z)$ rather than the objective itself. To produce structured skills that lead to diverse behaviors, $I(\tau; z)$ estimators must (i) explicitly encourage diverse behaviors and (ii) have the capacity to discriminate between high-dimensional continuous skills. Current approaches do not satisfy both criteria.

Competence-base algorithms do not ensure diverse behaviors: Most of the best known competence-based approaches [17, 18, 25, 26], optimize the first decomposition of the mutual information $\mathcal{H}(z) - \mathcal{H}(z|\tau)$. The issue with this decomposition is that while it ensures diversity of skill vectors it does not ensure diverse behavior from the policy, meaning $\max \mathcal{H}(z)$ does not imply $\max \mathcal{H}(\tau)$. Of course, if $H(z) - \mathcal{H}(z|\tau)$ is maximized and the skill dimension is sufficiently large, then $\mathcal{H}(\tau)$ will also be maximized implicitly. Yet in practice, to learn an accurate discriminator $q(z|\tau)$, the above methods assume skill spaces that are much smaller than the state space (see Table 2), and thus behavioral diversity may not be guaranteed. In contrast, the decomposition $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ ensures diverse behaviors through the entropy term $\mathcal{H}(\tau)$. Methods that utilize this decomposition include [27, 20].

Why it is important to utilize high-dimensional skills: Once a policy is capable of generating diverse behaviors, it is important that the discriminator can distill these behaviors into distinct skills. If the set of behaviors outnumbers the set of skills, this will result in degenerate skills – when one skill maps to multiple different behaviors. It is therefore important that the discriminator can accommodate continuous skills of sufficiently high dimension. Empirically, the discriminators used in prior work utilize only low-dimensional continuous skill vectors. DIAYN [17] utilized 16 dimensional skills, DADS [20] utilizes continuous skills of dimension $2 - 5$, while APS [27], an algorithm that utilizes successor features [30, 31] for the discriminator, is only capable of learning continuous skills with

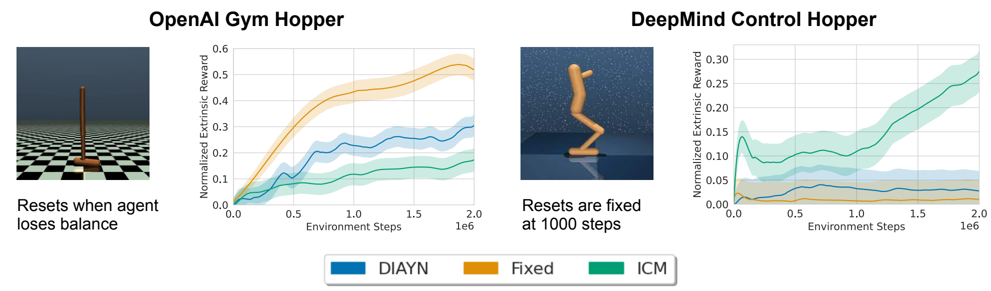

Figure 3: To empirically demonstrate issues inherent to competence-based exploration methods, we run DIAYN [17] and compare it to ICM [12] and a *Fixed* baseline where the agent receives an intrinsic reward of 1.0 for each timestep and no extrinsic reward on both OpenAI Gym *(episode resets when agent loses balance)* and DeepMind Control (DMC) *(episode is fixed for 1k steps)* Hopper environments. Since Gym and DMC rewards are on different scales, we normalize rewards based on the maximum reward achieved by any algorithm (1k for Gym, 3 for DMC). While DIAYN is able to achieve higher extrinsic rewards than ICM on Gym, the Fixed intrinsic reward baseline performs best. However, on DMC the Fixed and DIAYN agents achieve near-zero reward while ICM does not. This is consistent with findings of prior work that DIAYN is able to learn diverse behaviors in Gym [17] as well as the observation that DIAYN performs poorly on DMC environments [21]

dimension 10. We show how small skill spaces can lead to ineffective exploration in a simple gridworld setting in Appendix G and evidence that skill dimension affects performance in Fig. 5.

On the importance of benchmarks for evaluation: While prior competence-based approaches such as DIAYN [17] were evaluated on OpenAI Gym [32], Gym environment episodes terminate when the agent loses balance thereby leaking some aspects of extrinsic signal to the exploration agent. On the other hand, DMC episodes have fixed length. We show in Fig 3 that this small difference in environments results in large performance differences. Specifically, we find that DIAYN is able to learn diverse skills in Gym but not in DMC, which is consistent with both observations from DIAYN and URLB papers. Due to fixed episode lengths, DMC tasks are harder for reward-free exploration since agents must learn to balance without supervision.

3 Method

3.1 Contrastive Intrinsic Control

From Section 2 we are motivated to find a lower bound for $I(\tau; z)$ with a discriminator that is capable of supporting high-dimensional continuous skills[3]. Additionally, we wish to increase the diversity of behaviors so that the discriminator can continue learning new skills throughout training. We choose the forward decomposition of MI $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ similar to [27] and estimate the lower bound with Eq. 2. The entropy is estimated $\mathcal{H}(\tau)$ with a particle-based estimator similar to [15]. As such, the primary technical contribution of this work is a novel estimator for the discriminator.

To improve the discriminator, we propose to utilize noise contrastive estimation (NCE) [23] between state-transitions and latent skills as a lower bound for $I(\tau; z)$. It has been shown previously that such estimators provide a valid lower bound for mutual information [33]. However, to the best of our knowledge, this is the first work to investigate contrastive representation learning for intrinsic control.

Representation Learning: Specifically, we propose to learn embeddings by parameterizing the discriminator with a contrastive density estimator. This is a novel choice that differs from prior works which utilize a classifier [17] or non-contrastive density estimation [20].

$$\log q(\tau|z) := f(\tau, z) - \log \frac{1}{N} \sum_{j=1}^{N} \exp(f(\tau_j, z)). \tag{3}$$

where $f(\tau, z)$ is any real valued function.

[3]In high-dimensional state-action spaces the number of distinct behaviors can be quite large.

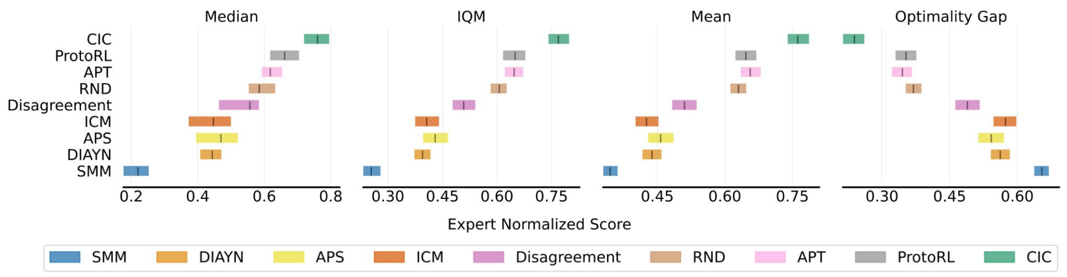

Figure 4: We report the aggregate statistics using stratified bootstrap intervals [35] for 12 downstream tasks on URLB with 10 seeds, so each statistic for each algorithm has 120 seeds in total. We find that overall, CIC achieves leading performance on URLB in terms of the IQM, mean, and OG statistics. As recommended by [35], we use the IQM as our primary performance measure. In terms of IQM, CIC improves upon the next best skill discovery algorithm (APS) by 79% and the next best algorithm overall (ProtoRL) by 18%.

For our practical algorithm, we parameterize this function as $f(\tau, z) = g_{\psi_1}(\tau)^\top g_{\psi_2}(z)/\|g_{\psi_1}(\tau)\|\|g_{\psi_2}(z)\|T$ where $\tau = (s, s')$ is a transition tuple, g_{ψ_k} are neural encoders, and T is a temperature parameter. This inner product is similar to the one used in SimCLR [34].

The representation learning loss backpropagates gradients from the NCE loss which maximizes similarity between state-transitions and corresponding skills.

$$F_{\text{CIC}}(\tau) = \frac{g_{\psi_1}(\tau_i)^\top g_{\psi_2}(z_i)}{\|g_{\psi_1}(\tau_i)\|\|g_{\psi_2}(z_i)\|T} - \log \frac{1}{N}\sum_{j=1}^{N} \exp\left(\frac{g_{\psi_1}(\tau_j)^\top g_{\psi_2}(z_i)}{\|g_{\psi_1}(\tau_j)\|\|g_{\psi_2}(z_i)\|T}\right) \tag{4}$$

where $N-1$ is the number of negatives. The total number of elements in the summation is N because it includes one positive, so the index j includes the positive index i similar to the objective in [33]. We provide pseudocode for the CIC representation learning loss:

```
"""
PyTorch-like pseudocode for the CIC loss
"""

def cic_loss(s, s_next, z, temp):
    """
    states: s, s_next (B, D), skills: z (B, D)
    """

    tau = concat(s, s_next, dim=1)

    query = query_net(z)
    key = key_net(tau)

    query = normalize(query, dim=1)
    key = normalize(key, dim=1)

    logits = matmul(query, key.T) / temp #logits are (B, B)
    labels = arange(logits.shape[0]) # positives are on diagonal

    # softmax_cross_entropy API same as in PyTorch docs
    loss = softmax_cross_entropy(logits, labels)

    return loss
```

Listing 1: Pseudocode for the CIC loss.

Intrinsic reward: Although we have a representation learning objective, we still need to specify the intrinsic reward for the algorithm for which there can be multiple choices. Prior works consider specifying an intrinsic reward that is proportional to state-transition entropy [15], the discriminator [17], a similarity score between states and skills [36], and the uncertainty of the discriminator [37].

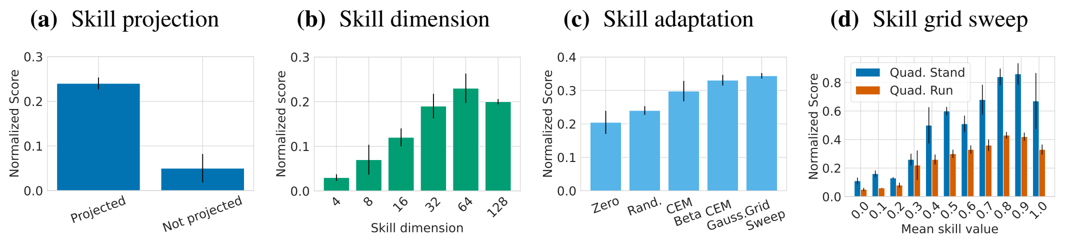

Figure 5: Design choices for pre-training and adapting with skills have significant impact on performance. In (a) and (b) the agent's zero-shot performance is evaluated while sampling skills randomly while in (c) and (d) the agent's performance is evaluated after finetuning the skills vector. *(a)* we show empirically that the projecting skill vectors after sampling them from noise significantly improves the agent's performance. *(b)* The skill dimension is a crucial hyperparameter and, unlike prior methods, CIC scales to large skill vectors achieving optimal performance at 64 dimensional skills. *(c)* We test several adapation strategies and find that a simple grid search performs best given the small 4k step adaptation budget, *(d)* Choosing the right skill vector has substantial impact on performance and grid sweeping allows the agent to select the appropriate skill.

We investigate each of these choices in Section 6 and find that an intrinsic reward that maximizes state-transition entropy coupled with representation learning via the CPC loss defined in Sec. 3.1 is the simplest variant that also performs well (see Table 1), which we use for all other experiments.

For the intrinsic reward, we use a particle estimate [22, 38] as in [15] of the state-transition entropy. Similar to [15, 16] we estimate the entropy up to a proportionality constant, because we want the agent to maximize entropy rather than estimate its exact value.

The APT particle entropy estimate is proportional to the distance between the current visited state transition and previously seen neighboring points.

$$\mathcal{H}_{particle}(\tau) \propto \frac{1}{N_k} \sum_{h_i^\star \in N_k}^{N_k} \log \|h_i - h_i^\star\| \tag{5}$$

where h_i is an embedding of τ_i shown in Fig. 2, h_i^* is a kNN embedding, N_k is the number of kNNs.

Explore and Exploit: With these design choices the two components of the CIC algorithm can be interpreted as *exploration* with intrinsic rewards and *exploitation* using representation learning to distill behaviors into skills. The marginal entropy maximizes the diversity of state-transition embeddings while the contrastive discriminator $\log q(\tau|z)$ encourages exploitation by ensuring that skills z lead to predictable states τ. Together the two terms incentivize the discovery of diverse yet predictable behaviors from the RL agent. While CIC shares a similar intrinsic reward structure to APT [15], we show that the new representation learning loss from the CIC estimator results in substantial performance gains in Sec 6.

4 Practical Implementation

Our practical implementation consists of two main components: the RL optimization algorithm and the CIC architecture. For fairness and clarity of comparison, we use the same RL optimization algorithm for our method and all baselines in this work. Since the baselines implemented in URLB [21] use a DDPG[4] [29] as their backbone, we opt for the same DDPG architecture to optimize our method as well (see Appendix B).

CIC Architecture: We use a particle estimator as in [15] to estimate $\mathcal{H}(\tau)$. To compute the variational density $q(\tau|z)$, we first sample skills from uniform noise $z \sim p(z)$ where $p(z)$ is the uniform distribution over the $[0, 1]$ interval. We then use two MLP encoders to embed $g_{\psi_1}(\tau)$ and $g_{\psi_2}(z)$,

[4] It was recently was shown that a DDPG achieves state-of-the-art performance [39] on DeepMind Control [40] and is more stable than SAC [41] on this benchmark.

and optimize the parameters ψ_1, ψ_2 with the CPC loss similar to SimCLR [34] since $f(\tau, z) = g_{\psi_1}(\tau)^T g_{\psi_2}(z)$. We fix the hyperparameters across all domains and downstream tasks. We refer the reader to the Appendices D and E for the full algorithm and a full list of hyperparameters.

Adapting to downstream tasks: To adapt to downstream tasks we follow the same procedure for competence-based method adaptation as in URLB [21]. During the first 4k environment interactions we populate the DDPG replay buffer with samples and use the extrinsic rewards collected during this period to finetune the skill vector z. While it's common to finetune skills with Cross Entropy Adaptation (CMA), given our limited budget of 4k samples (only 4 episodes) we find that a simple grid sweep of skills over the interval $[0, 1]$ produces the best results (see Fig. 5). After this, we fix the skill z and finetune the DDPG actor-critic parameters against the extrinsic reward for the remaining 96k steps. Note that competence-based methods in URLB also finetune their skills during the first 4k finetuning steps ensuring a fair comparison between the methods. The full adaptation procedure is detailed in Appendix D.

5 Experimental Setup

Environments We evaluate our approach on tasks from URLB, which consists of twelve downstream tasks across three challenging continuous control domains for exploration algorithms – walker, quadruped, and Jaco arm. Walker requires a biped constrained to a 2D vertical plane to perform locomotion tasks while balancing. Quadruped is more challenging due to a higher-dimensional state-action space and requires a quadruped to in a 3D environment to learn locomotion skills. Jaco arm is a 6-DOF robotic arm with a three-finger gripper to move and manipulate objects without locking. All three environments are challenging in the absence of an extrinsic reward.

Baselines: We implemented CIC using the URLB [21] codebase [5] and compare CIC to baselines included in URLB across all three exploration categories. Knowledge-based basedlines include ICM [12], Disagreement [13], and RND [14]. Data-based baselines incude APT [15] and ProtoRL [16]. Competence-based baselines include DIAYN [17], SMM [26], and APS [27]. The closest baselines to CIC are APT, which is similar to CIC but without state-skill CPC representation learning (no discriminator), and APS which uses the same decomposition of mutual information as CIC and also uses a particle entropy estimate for $\mathcal{H}(\tau)$. The main difference between APS and CIC is that APS uses successor features while CIC uses a contrastive estimator for the discriminator. For further details regarding baselines we refer the reader to Appendix C.

Evaluation: We follow an identical evaluation to the 2M pre-training setup in URLB. First, we pre-train each RL agent with the intrinsic rewards for 2M steps. Then, we finetune each agent to the downstream task with extrinsic rewards for 100k steps. All baselines were run for 10 seeds per downstream task for each algorithm using the code and hyperparameters provided by URLB [21]. Built on top of URLB, CIC is also run for 10 seeds per task. A total of $1080 = 9$ algorithms $\times$ 12 tasks $\times$ 10 seeds experiments were run for the main results. Importantly, all baselines and CIC use a DDPG agent as their backbone.

To ensure that our evaluation statistics are unbiased we use stratified bootstrap confidence intervals to report aggregate statistics across M runs with N seeds as described in *Rliable* [35] to report statistics for our main results in Fig. 4. Our primary success metric is the interquartile mean (IQM) and the Optimality Gap (OG). IQM discards the top and bottom 25% of runs and then computes the mean. It is less susceptible to outliers than the mean and was shown to be the most reliable statistic for reporting results for RL experiments in [35]. OG measures how far a policy is from optimal (expert) performance. To define expert performance we use the convention in URLB, which is the score achieved by a randomly initialized DDPG after 2M steps of finetuning (20x more steps than our finetuning budget).

6 Results

We investigate empirical answers to the following research questions: (Q1) How does CIC adaptation efficiency compare to prior competence-based algorithms and exploration algorithms more broadly?

[5]URLB is open-sourced under an MIT license `https://github.com/rll-research/url_benchmark/blob/main/LICENSE`.

(Q2) Which intrinsic reward instantiation of CIC performs best? (Q3) How do the two terms in the CIC objective affect algorithm performance? (Q4) How does skill selection affect the quality of the pre-trained policy? (Q5) Which architecture details matter most?

Adaptation efficiency of CIC and exploration baslines: Expert normalized scores of CIC and exploration algorithms from URLB are shown in Fig. 4. We find that CIC substantially outperforms prior competence-based algorithms (DIAYN, SMM, APS) achieving a 79% higher IQM than the next best competence-based method (APS) and, more broadly, achieving a 18% higher IQM than the next best overall baseline (ProtoRL). In further ablations, we find that the contributing factors to CIC's performance are its ability to accommodate substantially larger continuous skill spaces than prior competence-based methods.

Intrinsic reward specification: The intrinsic reward for competence-based algorithms can be instantiated in many different ways. Here, we analyze intrinsic reward for CIC with the form $r_{int} = H(\tau) + D(\tau, z)$, where D is some function of (τ, z). Prior works, select D to be (i) the discriminator [27], (ii) a cosine similarity between embeddings [36], (iii) uncertainty of the discriminator [37], and (iv) just the entropy $D(\tau, z) = 0$ [15]. We run CIC with each of these variants on the walker and quadruped tasks and measure the final mean performance across the downstream tasks (see Tab. 1). The results show that the entropy-only intrinsic reward performs best followed by an uncertainty-based intrinsic reward. We hypothesize that the reason why a simple entropy-only intrinsic reward works well is that state-skill CPC representation learning clusters similar behaviors together. Since similar behaviors are clustered, maximizing the entropy of state-transition embeddings produces increasingly diverse behaviors.

	disc.	similarity	uncertainty	entropy	APS
walker	0.79 ± 0.04	0.79 ± 0.03	0.76 ± 0.04	0.82 ± 0.02	$0.50 \pm .04$
quad.	0.45 ± 0.07	0.60 ± 0.05	0.70 ± 0.03	0.76 ± 0.03	$.48 \pm 0.02$
mean	0.62	0.70	0.73	0.80	0.49

Table 1: Comparing different potential intrinsic rewards for CIC, we find that entropy-based intrinsic reward performs best, suggesting that the CIC discriminator is primarily useful for representation learning. These are normalized scores averaged over 5 seeds across 8 downstream tasks. Note that all intrinsic reward specifications outperform the baseline methods. Since the particle entropy estimates a quantity proportional to the entropy, two-term intrinsic rewards need to be carefully balanced with a hyperparameter. We believe this is the reason the various intrinsic rewards perform worse than entropy-only one.

The importance of representation learning: To what extent does representation learning with the state-skill CIC loss affect the agent's exploration capability? To answer this question we train the CIC agent with the entropy intrinsic reward with and without the representation learning auxiliary loss for 2M steps. The zero-shot reward plotted in Fig. 6 indicates that without representation learning the policy collapses. With representation learning, the agent is able to discover diverse skills evidenced by the non-zero reward. This result suggests that state-skill CPC representation learning is a critical part of CIC.

Qualitative analysis of CIC behaviors: Qualitatively, we find that CIC is able to learn locomotion behaviors in DMC without extrinsic information such as early termination as in OpenAI Gym. While most skills are higher entropy and thus more chaotic, we show in Fig 1 that structured behaviors can be isolated by fixing a particular skill vector. For example, in the walker and quadruped domains - balancing, walking, and flipping skills can be isolated. For more qualitative investigations we refer the reader to Appendix H.

Skill architecture and adaptation ablations: We find that projecting the skill to a latent space before inputting it as the key for the contrastive loss is an important design decision (see Fig. 5a), most likely because this reduces the diversity of the skill vector making the discriminator task simpler.

We also find empirically that the skill dimension is an important hyperparameter and that larger skills results in better zero-shot performance (see Fig. 5b), which empirically supports the hypothesis posed in Section 2 and Appendix G that larger skill spaces are important for internalizing diverse behaviors. Interestingly, CIC zero-shot performance is poor in lower skill dimensions (e.g. $\dim(z) < 10$),

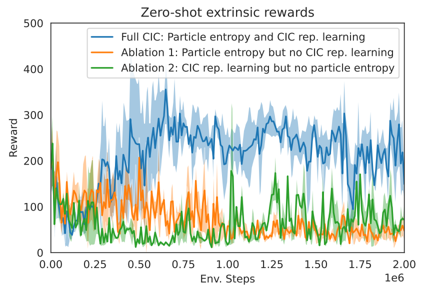

Figure 6: Mean zero-shot extrinsic rewards for Quadruped stand over 3 seeds with and without state-skill representation learning. Without representation learning, the algorithm collapses. Similarly, with CIC representation learning but no entropy term (in which case we use the discriminator as the intrinsic reward) the policy also collapses. Note that there is no finetuning happening here. We're showing the task-specific extrinsic reward during reward-free pre-training as a way to sense-check exploration policy.

suggesting that when dim(z) is small CIC performs no better than prior competence-based methods such as DIAYN, and that scaling to larger skills enables CIC to pre-train effectively.

To measure the effect of skill finetuning described in Section 4, we sweep mean skill values along the interval of the uniform prior $[0, 1]$ with a budget of 4k total environment interactions and read out the performance on the downstream task. By sweeping, we mean simply iterating over the interval $[0, 1]$ with fixed step size (e.g. $v = 0, 0.1, \ldots, 0.9, 1$) and setting $z_i = v$ for all i. This is not an optimal skill sampling strategy but works well due to the extremely limited number of samples for skill selection. We evaluate this ablation on the Quadruped Stand and Run downstream tasks. The results shown in Fig. 5 indicate that skill selection can substantially affect zero-shot downstream task performance.

7 Related Work

The most closely related prior algorithms to CIC are APT [15] and APS [27]. Both CIC and APS use the $\mathcal{H}(\tau) - \mathcal{H}(\tau|z)$ decomposition of the mutual information and both used a particle estimator [22] to compute the state entropy as in [15]. The main difference between CIC and APS is the discriminator. APS uses successor features as in [31] for its discriminator while CIC uses a noise contrastive estimator. Unlike successor features, which empirically only accommodate low-dimensional continuous skill spaces (see Table 2), the noise contrastive discriminator is able to leverage higher continuous dimensional skill vectors. Like APT, CIC has an intrinsic reward that maximizes $\mathcal{H}(\tau)$. However, CIC also does contrastive skill learning to shape the embedding space and outputs a skill-conditioned policy.

The CIC discriminator is similar to the one used in DISCERN [36], a goal-conditioned unsupervised RL algorithm. Both methods use a contrastive discriminator by sampling negatives and computing an inner product between queries and keys. The main differences are (i) that DISCERN maximizes $I(\tau; g)$ where g are image goal embeddings while CIC maximizes $I(\tau; z)$ where z are abstract skill vectors; (ii) DISCERN uses the DIAYN-style decomposition $I(\tau; g) = H(g) - H(g|\tau)$ while CIC decomposes through $H(\tau) - H(\tau|z)$, and (iii) DISCERN discards the $H(g)$ term by sampling goals uniformly while CIC explicitly maximizes $\mathcal{H}(\tau)$. While DISCERN and CIC share similarities, DISCERN operates over image goals while CIC operates over abstract skill vectors so the two methods are not directly comparable.

Finally, another similar algorithm to CIC is DADS [20] which also decomposes through $H(\tau) - H(\tau|z)$. While CIC uses a contrastive density estimate for the discriminator, DADS uses a maximum likelihood estimator similar to DIAYN. DADS maximizes $I(s'|s, z)$ and estimates entropy $\mathcal{H}(s'|s)$ by marginalizing over z such that $\mathcal{H}(s'|s) = -\log \sum_i q(s'|s, z_i)$ while CIC uses a particle estimator.

Table 2: Competence-based Unsupervised Skill Discovery Algorithms

Algorithm	Intrinsic Reward	Decomposition	Explicit $\max \mathcal{H}(\tau)$	Skill Dim.	Skill Space
SSN4HRL [42]	$\log q_\psi(z\|s_t)$	$H(z) - H(z\|\tau)$	No	6	discrete one-hot
VIC [18]	$\log q_\psi(z\|s_H))$	$H(z) - H(z\|\tau)$	No	60	discrete one-hot
VALOR [25]	$\log q_\psi(z\|s_{1:H})$	$H(z) - H(z\|\tau)$	No	64	discrete one-hot
DIAYN [17]	$\log q_\psi(z\|s_t)$	$H(z) - H(z\|\tau)$	No	128	discrete one-hot
DADS [20]	$q_\psi(s'\|z,s) - \sum_i \log q(s'\|z_i,s)$	$H(\tau) - H(\tau\|z)$	Yes	5	continuous
VISR [31]	$\log q_\psi(z\|s_t)$	$H(z) - H(z\|\tau)$	No	10	continuous
APS [27]	$F_{\text{Successor}}(s\|z) + \mathcal{H}_{\text{particle}}(s)$	$\mathcal{H}(\tau) - \mathcal{H}(\tau\|z)$	Yes	10	continuous
CIC	$F_{\text{CIC}}(s,s'\|z) + \mathcal{H}_{\text{particle}}(s,s')$	$\mathcal{H}(\tau) - \mathcal{H}(\tau\|z)$	Yes	64	continuous

Table 3: A list of competence-based algorithms. We describe the intrinsic reward optimized by each method and the decomposition of the mutual information utilized by the method. We also note whether the method explicitly maximizes state transition entropy. Finally, we note the maximal dimension used in each work and whether the skills are discrete or continuous. All methods prior to CIC only support small skill spaces, either because they are discrete or continuous but low-dimensional.

8 Limitations and Impact

While CIC achieves leading results on state-based URLB, we would also like to address its limitations. First, in this paper we only consider MDPs (and not partially observed MDPs) where the full state is observable. We focus on MDPs because generating diverse behaviors in environments with large state spaces has been the primary bottleneck for competence-based exploration. Combining CIC with visual representation learning to scale this method to pixel-based inputs is a promising future direction for research not considered in this work.

One issue with unsupervised RL algorithms (and hence CIC) in terms of potentially negative societal impact is that self-supervised exploration can be dangerous. Since self-supervised agents maximize intrinsic rewards, this can lead to destructive behavior. For example, when deploying CIC on a Walker or Quadruped robot it learns chaotic exploration behaviors [6] that would most likely break the robot in real-world settings. Alignment of exploration agents to prevent them from learning dangerous policies is a promising direction for future work.

9 Conclusion

We have introduced a new competence-based algorithm – Contrastive Intrinsic Control (CIC) – which enables more effective exploration than prior unsupervised skill discovery algorithms by explicitly encouraging diverse behavior while distilling predictable behaviors into skills with a contrastive discriminator. We showed that CIC is the first competence-based approach to achieve leading performance on URLB. We hope that this encourages further research in developing RL agents capable of generalization.

10 Acknowledgements

We would like to thank Ademi Adeniji, Xinyang Geng, Fangchen Liu for helpful discussions. We would also like to thank Phil Bachman for useful feedback. This work was partially supported by Berkeley DeepDrive, NSF AI4OPT AI Institute for Advances in Optimization under NSF 2112533, and the Office of Naval Research grant N00014-21-1-2769.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[6] See `https://sites.google.com/view/cicneurips2022/home`

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[3] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Artfhur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

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

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

[6] Xue Bin Peng, P. Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37:143:1–143:14, 2018.

[7] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[8] OpenAI. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[9] OpenAI. Solving rubik's cube with a robot hand. *ArXiv*, abs/1910.07113, 2019.

[10] Aravind Srinivas and Pieter Abbeel. Unsupervised learning for reinforcement learning, 2021.

[11] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

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

[13] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, 2019.

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

[15] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021.

[16] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, 2021.

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

[18] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

[19] Taehwan Kwon. Variational intrinsic control revisited. In *International Conference on Learning Representations*, 2021.

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

[21] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark, 2021.

[22] Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23(3-4):301–321, 2003.

[23] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[25] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[26] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *CoRR*, abs/1906.05274, 2019.

[27] Hao Liu and Pieter Abbeel. APS: active pretraining with successor features. In *International Conference on Machine Learning*, 2021.

[28] David Barber and Felix V. Agakov. The im algorithm: A variational approach to information maximization. In *Advances in neural information processing systems*, 2003.

[29] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[30] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

[31] Steven Hansen, Will Dabney, André Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2020.

[32] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[33] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[34] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020.

[35] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice, 2021.

[36] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards, 2018.

[37] DJ Strouse, Kate Baumli, David Warde-Farley, Vlad Mnih, and Steven Hansen. Learning more skills through optimistic exploration. *CoRR*, abs/2107.14226, 2021.

[38] J Beirlant. Nonparametric entropy estimation: An overview. *International Journal of the Mathematical Statistics Sciences*, 6:17–39, 1997.

[39] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021.

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

[41] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[42] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2018.

[43] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020.

[44] Tom Zahavy, Andre Barreto, Daniel J Mankowitz, Shaobo Hou, Brendan O'Donoghue, Iurii Kemaev, and Satinder Baveja Singh. Discovering a set of policies for the worst case reward, 2021.

[45] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

[46] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180. PMLR, 2019.

[47] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

A Competence-based Exploration Algorithms

The competence-based algorithms considered in this work aim to maximize $I(\tau; s)$. The algorithms differ by ho they decompose mutual information, whether they explicitly maximize behavioral entropy, their skill space (discrete or continuous) and their intrinsic reward structure. We provide a list of common competence-based algorithms in Table 2.

B Deep Deterministic Policy Gradient (DDPG)

A DDPG is an actor-critic RL algorithm that performs off-policy gradient updates and learns a Q function $Q_\phi(s, a)$ and an actor $\pi_\theta(a|s)$. The critic is trained by satisfying the Bellman equation.

$$\mathcal{L}_Q(\phi, \mathcal{D}) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\Big(Q_\phi(s_t, a_t) - r_t - \gamma Q_{\bar{\phi}}(s_{t+1}, \pi_\theta(s_{t+1})\Big)^2\right]. \tag{6}$$

Here, $\bar{\phi}$ is the Polyak average of the parameters ϕ. As the critic minimizes the Bellman error, the actor maximizes the action-value function.

$$\mathcal{L}_\pi(\theta, \mathcal{D}) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[Q_\phi(s_t, \pi_\theta(s_t))\right]. \tag{7}$$

C Baselines

For baselines, we choose the existing set of benchmarked unsupervised RL algorithms on URLB. We provide a quick summary of each method. For more detailed descriptions of each baseline we refer the reader to URLB [21]

Competence-based Baselines: CIC is a competence-based exploration algorithm. For baselines, we compare it to DIAYN [17], SMM [26], and APS [27]. Each of these algorithms is described in Table 2. Notably, APS is a recent state-of-the-art competence-based method that is the most closely related algorithm to the CIC algorithm.

Knowledge-based Baselines: For knowledge-based baselines, we compare to ICM [12], Disagreement [13], and RND [14]. ICM and RND train a dynamics model and random network prediction model and define the intrinsic reward to be proportional to the prediction error. Disagreement trains an ensemble of dynamics models and defines the intrinsic reward to be proportional to the uncertainty of an ensemble.

Data-based Baselines: For data-based baselines we compare to APT [15] and ProtoRL [16]. Both methods use a particle estimator to estimate the state visitation entropy. ProtoRL also performs discrete contrastive clustering as in [43] as an auxiliary task and uses the resulting clusters to compute the particle entropy. While ProtoRL is more effective than APT when learning from pixels, on state-based URLB APT is competitive with ProtoRL. Our method CIC is effectively a skill-conditioned APT agent with a contrastive discriminator.

D Full CIC Algorithm

The full CIC algorithm with both pre-training and fine-tuning phases is shown in Algorithm 1. We pre-train CIC for 2M steps, and finetune it on each task for 100k steps.

Algorithm 1 `Contrastive Intrinsic Control`

```
Require: Initialize all networks: encoders $g_{\psi_1}$ and $g_{\psi_2}$, actor $\pi_\theta$, critic $Q_\phi$, replay buffer $\mathcal{D}$.
Require: Environment (env), $M$ downstream tasks $T_k, k \in [1, \dots, M]$.
Require: pre-train $N_{\text{PT}} = 2M$ and fine-tune $N_{\text{FT}} = 100K$ steps.
 1: for $t = 1..N_{\text{PT}}$ do                                        ▷ Part 1: Unsupervised Pre-training
 2:     Sample and encode skill $z \sim p(z)$ and $z \leftarrow g_{\psi_2}(z)$
 3:     Encode state $s_t \leftarrow g_{\psi_1}(s_t)$ and sample action $a_t \leftarrow \pi_\theta(s_t, z) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$
 4:     Observe next state $s_{t+1} \sim P(\cdot|s_t, a_t)$
 5:     Add transition to replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, s_{t+1})$
 6:     Sample a minibatch from $\mathcal{D}$, compute contrastive loss in Eq.4 and update encoders $g_{\psi_1}, g_{\psi_2}$, compute
    CIC intrinsic reward with Eq. 5 and update actor $\pi_\theta$ and critic $Q_\phi$
 7: end for
 8: for $T_k \in [T_1, \dots, T_M]$ do                                   ▷ Part 2: Supervised Fine-tuning
 9:     Initialize all networks with weights from pre-training phase and an empty replay buffer $\mathcal{D}$.
10:     for $t = 1 \dots 4,000$ do
11:         Take random action $a_t \sim \mathcal{N}(0, 1)$
12:         Select skill with grid sweep over unit interval $[0, 1]$ every 100 steps
13:         Sample minibatch from $\mathcal{D}$ and update actor $\pi_\theta$ and critic $Q_\phi$
14:     end for
15:     Fix skill $z$ that achieved highest extrinsic reward during grid sweep.
16:     for $t = 4,000 \dots N_{\text{FT}}$ do
17:         Encode state $s_t \leftarrow g_{\psi_1}(s_t)$ and sample action $a_t \leftarrow \pi_\theta(s_t, z) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$
18:         Observe next state and reward $s_{t+1}, r_t^{\text{ext}} \sim P(\cdot|s_t, a_t)$
19:         Add transition to replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r_t^{\text{ext}}, s_{t+1})$
20:         Sample minibatch from $\mathcal{D}$ and update actor $\pi_\theta$ and critic $Q_\phi$.
21:     end for
22:     Evaluate performance of RL agent on task $T_k$
23: end for
```

E Hyper-parameters

Baseline hyperparameters are taken from URLB [21], which were selected by performing a grid sweep over tasks and picking the best performing set of hyperparameters. Except for the skill dimension, hyperparameters for CIC are borrowed from URLB.

Table 4: Hyper-parameters used for CIC .

DDPG hyper-parameter	Value
Replay buffer capacity	10^6
Action repeat	1
Seed frames	4000
n-step returns	3
Mini-batch size	1024
Seed frames	4000
Discount (γ)	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Critic target EMA rate (τ_Q)	0.01
Features dim.	1024
Hidden dim.	1024
Exploration stddev clip	0.3
Exploration stddev value	0.2
Number pre-training frames	2×10^6
Number fine-turning frames	1×10^5
CIC hyper-parameter	**Value**
Skill dim	64 continuous
Prior	Uniform [0,1]
Skill sampling frequency (steps)	50
State net arch. $g_{\psi_1}(s)$	$\dim(\mathcal{O}) \to 1024 \to 1024 \to 64$ ReLU MLP
Skill net arch. $g_{\psi_2}(z)$	$64 \to 1024 \to 1024 \to 64$ ReLU MLP
Prediction net arch.	$64 \to 1024 \to 1024 \to 64$ ReLU MLP

Statistic	ICM	Dis.	RND	APT	Proto	DIAYN	APS	SMM	CIC	% CIC > APS	% CIC > Proto
Median ↑	0.45	0.56	0.58	0.62	0.66	0.44	0.47	0.22	0.76	+61%	+15%
IQM ↑	0.41	0.51	0.61	0.65	0.65	0.40	0.43	0.25	0.77	+79%	+18%
Mean ↑	0.43	0.51	0.63	0.66	0.65	0.44	0.46	0.35	0.76	+65%	+17%
OG ↓	0.57	0.49	0.37	0.35	0.35	0.56	0.54	0.65	0.24	-44%	-68%

Table 5: Statics for downstream task normalized scores for CIC and baselines from URLB [21]. CIC improves over both the prior leading competence-based method APS [27] and overall next-best exploration algorithm ProtoRL [16] across all readout statistics. Each data point is a statistic computed using 10 seeds and 12 downstream tasks (120 experiments per data point). The statistics are computed using RLiable [35].

F Raw Numerical Results

We provide a list of raw numerical results for finetuning CIC and baselines in Tables 5 and 6. All baselines were run using the code provided by URLB [21] for 10 seeds per downstream task.

					Pre-training for 2×10^6 environment steps							
Domain	Task	Expert	DDPG	CIC	ICM	Disagreement	RND	APT	ProtoRL	SMM	DIAYN	APS
Walker	Flip	799	538±27	631 ± 34	417±16	346±13	474±39	544±14	456±12	450±24	319±17	465±20
	Run	796	325±25	486 ± 25	247±21	208±15	406±30	392±26	306±13	426±26	158±8	134±16
	Stand	984	899±23	959 ± 2	859±23	746±34	911±5	942±6	917±27	924±12	695±46	721±44
	Walk	971	748±47	885 ± 28	627±42	549±37	704±30	773±70	792±41	770±44	498±27	527±79
Quadruped	Jump	888	236±48	595 ± 42	178±35	389±62	637±12	648±18	617±44	96±7	660±43	463±51
	Run	888	157±31	505 ± 47	110±18	337±30	459±6	492±14	373±33	96±6	433±29	281±17
	Stand	920	392±73	761 ± 54	312±68	512±89	766±43	872±23	716±56	123±11	851±43	542±53
	Walk	866	229±57	723 ± 43	126±27	293±37	536±39	770±47	412±54	80±6	576±81	436±79
Jaco	Reach bottom left	193	72±22	138 ± 9	111±11	124±7	110±5	103±8	129±8	45±7	39±6	76±8
	Reach bottom right	203	117±18	145 ± 7	97±9	115±10	117±7	100±6	132±8	46±11	38±5	88±11
	Reach top left	191	116±22	153 ± 7	82±14	106±12	99±6	73±12	123±9	36±3	19±4	68±6
	Reach top right	223	94±18	163 ± 4	103±11	139±7	100±6	90±10	159±7	47±6	28±6	76±10

Table 6: Performance of CIC and baselines on state-based URLB after first pre-training for 2×10^6 steps and then finetuning with extrinsic rewards for 1×10^5. All baselines were run for 10 seeds per downstream task for each algorithm using the code provided by URLB [21]. A total of $1080 = 9$ algorithms $\times$ 12 tasks $\times$ 10 seeds experiments were run.

G Toy Example to Illustrate the Need for Larger Skill Spaces

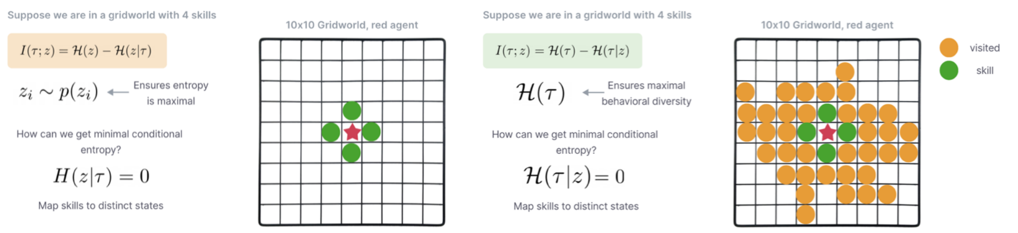
Suppose we are in a gridworld with 4 skills
$I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau)$
$z_i \sim p(z_i)$
Ensures entropy is maximal
How can we get minimal conditional entropy?
$H(z|\tau) = 0$
Map skills to distinct states
10x10 Gridworld, red agent
Suppose we are in a gridworld with 4 skills
$I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$
$\mathcal{H}(\tau)$
Ensures maximal behavioral diversity
How can we get minimal conditional entropy?
$\mathcal{H}(\tau|z) = 0$
Map skills to distinct states
10x10 Gridworld, red agent
visited
skill

Figure 7: A gridworld example motivating the need for large skill spaces. In this environment, we place an agent in a 10×10 gridworld and provide the agent access to four discrete skills. We show that the mutual information objective can be maximized by mapping these four skills to the nearest neighboring states resulting in low behavioral diversity and exploring only four of the hundred available states.

We illustrate the need for larger skill spaces with a gridworld example. Suppose we have an agent in a 10×10 sized gridworld and that we have four discrete skills at our disposal. Now let $\tau = s$ and consider how we may achieve maximal $I(\tau; z)$ in this setting. If we decompose $I(\tau; z) = \mathcal{H}(z) - \mathcal{H}(z|\tau)$ then we can achieve maximal $\mathcal{H}(z)$ by sampling the four skills uniformly $z \sim p(z)$. We can achieve $\mathcal{H}(z|\tau) = 0$ by mapping each skill to a distinct neighboring state of the agent. Thus, our mutual information is maximized but as a result the agent only explores four out of the hundrend available states in the gridworld.

Now suppose we consider the second decomposition $I(\tau; z) = \mathcal{H}(\tau) - \mathcal{H}(\tau|z)$. Since the agent is maximizing $\mathcal{H}(\tau)$ it is likely to visit a diverse set of states at first. However, as soon as it learns an accurate discriminator we will have $\mathcal{H}(\tau|z)$ and again the skills can be mapped to neighboring states to achieve minimal conditional entropy. As a result, the skill conditioned policy will only be able to reach four out of the hundrend possible states in this gridworld. This argument is shown visually in Fig. 7.

Skill spaces that are too large can also be an issue. Consider if we had 100 skills at our disposal in the same gridworld. Then the agent could minimize the conditional entropy by mapping each skill to a unique state which would result in the agent memorizing the environment by finding a one-to-one mapping between states and skills. While this is a potential issue it has not been encountered in practice yet since current competence-based methods support small skill spaces relative to the observation space of the environment.

H Qualitative Analysis of Skills

We provide two additional qualitative analyses of behaviors learned with the CIC algorithm. First, we take a simple pointmass setting and set the skill dimension to 1 in order to ablate the skills learned by the CIC agent in a simple setting. We sweep over different values of z and plot the behavioral flow vector field (direction in which point mass moves) in Fig.8. We find that the pointmass learns skills that produce continuous motion and that the direction of the motion changes as a function of the skill value. Near the origin the pointmass learns skills that span all directions, while near the edges the point mass learns to avoid wall collisions. Qualitatively, many behaviors are periodic.

Qualitatively, we find that methods like DIAYN that only support low dimensional skill vectors and do not explicitly incentivize diverse behaviors in their objective produce policies that map skills to a small set of static behaviors. These behaviors shown in Fig. 9 are non-trivial but also have low behavioral diversity and are not particularly useful for solving the downstream task. This observation is consistent with [44] where the authors found that DIAYN maps to static "yoga" poses in DeepMind Control. In contrast, behaviors produce by CIC are dynamic resulting flipping, jumping, and locomotive behaviors that can then be adapted to efficiently solve downstream tasks.

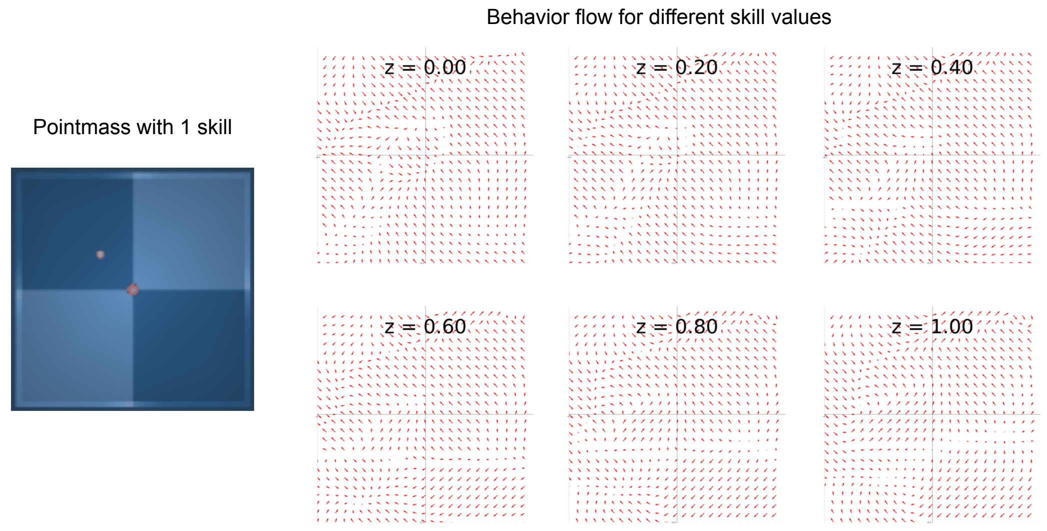

Figure 8: Learning curves for finetuning pre-trained agents for 100k steps. Task performance is aggregated for each domain, such that each curve represents the mean normalized scores over $4 \times 10 = 40$ seeds. The shaded regions represent the standard error. CIC surpasses the performance of the prior state-of-the-art on Walker and Jaco tasks while tying on Quadruped. CIC is the only algorithm that performs consistently well across all three domains.

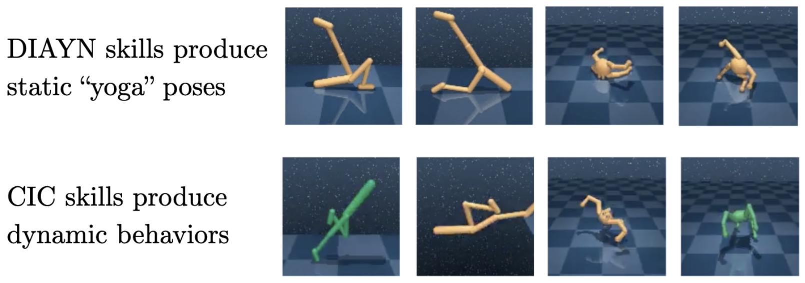
DIAYN skills produce static "yoga" poses
CIC skills produce dynamic behaviors

Figure 9: Qualitative visualization of DIAYN and CIC pre-training on the Walker and Quadruped domains from URLB. Confirming findings in prior work [44], we also find that DIAYN policies produce static but non-trivial behaviors mapping to "yoga" poses while CIC produces diverse and dynamic behaviors such as walking, flipping, and standing. Though it's hard to see from these images, all the DIAYN skills get stuck in frozen poses while the CIC skills are producing dynamic behavior with constant motion.

I OpenAI Gym vs. DeepMind control: How Early Termination Leaks Extrinsic Signal

Prior work on unsupervised skill discovery for continuous control [17, 20] was evaluated on OpenAI Gym [32] and showed diverse exploration on Gym environments. However, Gym environment episodes terminate early when the agent loses balance, thereby leaking information about the extrinsic task (e.g. balancing or moving). However, DeepMind Control (DMC) episodes have a fixed length of 1k steps. In DMC, exploration is therefore harder since the agent needs to learn to balance without any extrinsic signal.

To evaluate whether the difference in the two environments has impact on competence-based exploration, we run DIAYN on the hopper environments from both Gym and DMC. We compare to ICM, a popular exploration baseline, and a Fixed baseline where the agent receives an intrinsic reward of 1 for each timestep and no algorithms receive extrinsic rewards. We then measure the extrinsic reward, which loosely corresponds to the diversity of behaviors learned. Our results in Fig. 3 show that indeed DIAYN is able to learn diverse behaviors in Gym but not in DMC while ICM is able to learn diverse behaviors in both environments. Interestingly, the Fixed baseline achieves the highest reward on the Gym environment by learning to stand and balance. These results further motivate us to evaluate on URLB which is built on top of DMC.

J CIC vs Other Types of Contrastive Learning for RL

Contrastive learning in CIC is different than prior vision-based contrastive learning in RL such as CURL [45], since we are not performing contrastive learning over augmented images but rather over state transitions and skills. The contrastive objective in CIC is used for unsupervised learning of behaviors while in CURL it is used for unsupervised learning of visual features.

We provide pseudocode for the CIC loss below:

```
def discriminator_loss(states, next_states, skills, temp):
    """
    - states and skills are sampled from replay buffer
    - skills were sampled from uniform dist [0,1] during agent rollout
    - states / next_states: dim (B, D_state)
    - skills: dim (B, D_skill)
    """

    transitions = concat(states, next_states, dim=1)

    query = skill_net(skills) # (B, D_hidden) -> (B, D_hidden)
    key = transition_net(transitions) # (B, 2*D_state) -> (B, D_hidden
    )

    query = normalize(query, dim=1)
    key = normalize(key, dim=1)

    logits = matmul(query, key.T) / temp # (B, B)
    labels = arange(logits.shape[0])

    # positives are on diagonal, negatives are off diagonal
    # for each skill, negatives are sampled from transitions
    # while skills are fixed
    loss = cross_entropy(logits, labels)

    return loss
```

Listing 2: CIC discriminator loss

This is substantially different from prior contrastive learning works in RL such as CURL [45], which perform contrastive learning over images.

```
def curl_loss(obs, W, temp):
    """
    - observation images are sampled from replay buffer
    - obs: dim (B, C, H, W)
    - W: projection matrix (D_hidden, D_hidden)
    """

    query = aug(obs)
    key = aug(obs)

    query = cnn_net(query) # (B, D_hidden)
    key = cnn_net(key) # (B, D_hidden)

    logits = matmul(matmul(query, W), key.T) / temp # (B, B)
    labels = arange(logits.shape[0])

    # positives are on diagonal
    # negatives are off diagonal
    loss = cross_entropy(logits, labels)

    return loss
```

Listing 3: CURL contrastive loss

K On estimates of Mutual Information

In this work we have presented CIC - a new competence-based algorithm that achieves leading performance on URLB compared to prior unsupervised RL methods.

One might wonder whether estimating the exact mutual information (MI) or maximizing the tightest lower bound thereof is really the goal for unsupervised RL. In unsupervised representation learning, state-of-the-art methods like CPC and SimCLR maximize the lower bound of MI based on Noise Contrastive Estimation (NCE). However, as proven in CPC [33] and illustrated in [46] NCE is upper bounded by $\log N$, meaning that the bound is loose when the MI is larger than $\log N$. Nevertheless, these methods have been repeatedly shown to excel in practice. In [47] the authors show that the effectiveness of NCE results from the inductive bias in both the choice of feature extractor architectures and the parameterization of the employed MI estimators.

We have a similar belief for unsupervised RL - that with the right parameterization and inductive bias, the MI objective will facilitate behavior learning in unsupervised RL. This is why CIC lower bounds MI with (i) the particle based entropy estimator to ensure explicit exploration and (ii) a contrastive conditional entropy estimator to leverage the power of contrastive learning to discriminate skills. As demonstrated in our experiments, CIC outperforms prior methods, showing the effectiveness of optimizing an intrinsic reward with the CIC MI estimator.

L Compute Resources

CIC runs on a single GPU. In our experiments we used 4 NVIDIA TITAN RTX GPUs. Pre-training one seed of CIC for 2M steps takes 12-24 hours while fine-tuning to downstream tasks for 100k steps takes 30min - 1 hour.