Contrastive Learning as Goal-Conditioned Reinforcement Learning

Benjamin Eysenbach[α,β] **Tianjun Zhang**[γ] **Sergey Levine**[β,γ] **Ruslan Salakhutdinov**[α]
[α]CMU [β]Google Research [γ]UC Berkeley

Abstract

In reinforcement learning (RL), it is easier to solve a task if given a good representation. While *deep* RL should automatically acquire such good representations, prior work often finds that learning representations in an end-to-end fashion is unstable and instead equip RL algorithms with additional representation learning parts (e.g., auxiliary losses, data augmentation). How can we design RL algorithms that directly acquire good representations? In this paper, instead of adding representation learning parts to an existing RL algorithm, we show (contrastive) representation learning methods can be cast as RL algorithms in their own right. To do this, we build upon prior work and apply contrastive representation learning to action-labeled trajectories, in such a way that the (inner product of) learned representations exactly corresponds to a goal-conditioned value function. We use this idea to reinterpret a prior RL method as performing contrastive learning, and then use the idea to propose a much simpler method that achieves similar performance. Across a range of goal-conditioned RL tasks, we demonstrate that contrastive RL methods achieve higher success rates than prior non-contrastive methods, including in the offline RL setting. We also show that contrastive RL outperforms prior methods on image-based tasks, without using data augmentation or auxiliary objectives. [1]

1 Introduction

Representation learning is an integral part of reinforcement learning (RL[2]) algorithms. While such representations might emerge from end-to-end training [7, 79, 119, 126], prior work has found it necessary to equip RL algorithms with perception-specific loss functions [32, 44, 71, 89, 91, 101, 116, 140] or data augmentations [69, 73, 116, 118], effectively decoupling the representation learning problem from the reinforcement learning problem. Given what prior work has shown about RL in the presence of function approximation and state aliasing [2, 135, 138], it is not surprising that end-to-end learning of representations is fragile [69, 73]: an algorithm needs good representations to drive the learning of the RL algorithm, but the RL algorithm needs to drive the learning of good representations. So, *can we design RL algorithms that do learn good representations without the need for auxiliary perception losses?*

Rather than using a reinforcement learning algorithm also to solve a representation learning problem, we will use a representation learning algorithm to also solve certain types of reinforcement learning problems, namely goal-conditioned RL. Goal-conditioned RL is widely studied [6, 15, 22, 62, 80, 120], and intriguing from a representation learning perspective because it can be done in an entirely self-supervised manner, without manually-specified reward functions. We will focus on contrastive (representation) learning methods, using observations from the same trajectory (as done

[1]Project website with videos and code: `https://ben-eysenbach.github.io/contrastive_rl`
[2]RL = reinforcement learning, not representation learning.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

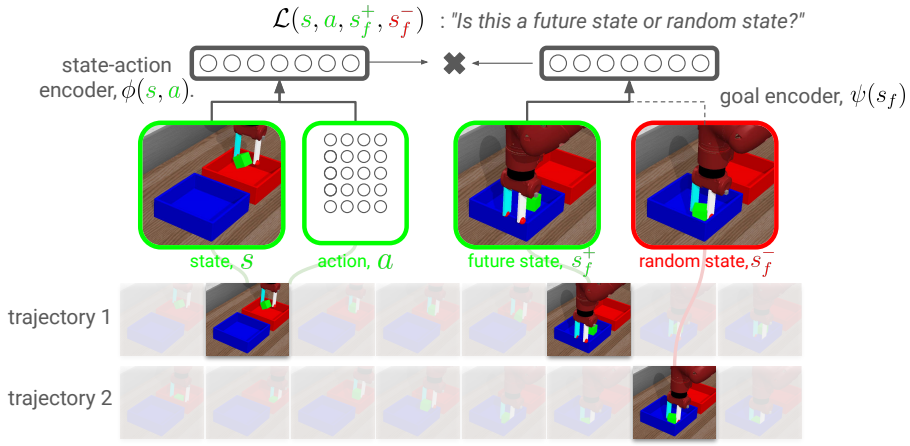


Figure 1: **Reinforcement learning via contrastive learning.** Our method uses contrastive learning to acquire representations of state-action pairs ($\phi(s, a)$) and future states ($\psi(s_f)$), so that the representations of future states are closer than the representations of random states. We prove that learned representation corresponds to a value function for a certain reward function. To select actions for reaching goal s_g, the policy chooses the action where $\phi(s, a)$ is closest to $\psi(s_g)$.

in prior work [95, 109]) while also including actions as an additional input (See Fig. 1). Intuitively, contrastive learning then resembles a goal-conditioned value function: nearby states have similar representations and unreachable states have different representations. We make this connection precise, showing that sampling positive pairs using the discounted state occupancy measure results in learning representations whose inner product exactly corresponds to a value function.

In this paper, we show how contrastive representation learning can be used to perform goal-conditioned RL. We formally relate the learned representations to reward maximization, showing that the inner product between representations corresponds to a value function. This framework of contrastive RL generalizes prior methods, such as C-learning [29], and suggests new goal-conditioned RL algorithms. One new method achieves performance similar to prior methods but is simpler; another method consistently outperforms the prior methods. On goal-conditioned RL tasks with image observations, contrastive RL methods outperform prior methods that employ data augmentation and auxiliary objectives, and do so without data augmentation or auxiliary objectives. In the offline setting, contrastive RL can outperform prior methods on benchmark goal-reaching tasks, sometimes by a wide margin.

2 Related Work

This paper will draw a connection between RL and contrastive representation learning, building upon a long line of contrastive learning methods in NLP and computer vision, and deep metric learning [17, 53, 54, 54, 56, 77, 84, 86, 87, 94, 95, 108, 109, 113, 122, 129, 132]. Contrastive learning methods learn representations such that similar ("positive") examples have similar representations and dissimilar ("negative") examples have dissimilar representations.[3] While most methods generate the "positive" examples via data augmentation, some methods generate similar examples using different camera viewpoints of the same scene [109, 122], or by sampling examples that occur close in time within time series data [4, 95, 109, 118]. Our analysis will focus on this latter strategy, as the dependence on time will allow us to draw a precise relationship with the time dependence in RL.

Deep RL algorithms promise to automatically learn good representations, in an end-to-end fashion. However, prior work has found it challenging to uphold this promise [7, 79, 119, 126], prompting many prior methods to employ separate objectives for representation learning and RL [32, 44, 71, 89, 91, 100, 101, 116, 118, 140, 143]. Many prior methods choose a representation learning objectives that reconstruct the input state [32, 47, 49, 50, 71, 91, 93, 141] while others use contrastive representation learning methods [89, 95, 111, 116, 118]. Unlike these prior methods, we will not use a separate representation learning objective, but instead use the same objective for both representation learning and reinforcement learning. Some prior RL methods have also used contrastive learning to acquire reward functions [14, 20, 33, 38, 63, 67, 92, 133, 134, 146], often in imitation learning

[3] Our focus will not be on recent methods that learn representations without negative samples [18, 43].

settings [37, 55]. In contrast, we will use contrastive learning to directly acquire a value function, which (unlike a reward function) can be used directly to take actions, without any additional RL.

This paper will focus on goal-conditioned RL problems, a problem prior work has approached using temporal difference learning [6, 29, 62, 80, 103, 106], conditional imitation learning [22, 41, 83, 105, 120], model-based methods [23, 107], hierarchical RL [90], and planning-based methods [30, 93, 105, 115]. The problems of automatically sampling goals and exploration [24, 35, 85, 98, 144] are orthogonal to this work. Like prior work, we will parametrize the value function as an inner product between learned representations [34, 58, 106]. Unlike these prior methods, we will learn a value function directly via contrastive learning, without using reward functions or TD learning.

Our analysis will be most similar to prior methods [11, 15, 29, 103] that view goal-conditioned RL as a data-driven problem, rather than as a reward-maximization problem. Many of these methods employ hindsight relabeling [6, 26, 62, 78], wherein experience is *relabeled* with an outcome that occurred in the future. Whereas hindsight relabeling is typically viewed as a trick to add on top of an RL algorithm, this paper can roughly be interpreted as showing that the hindsight relabeling is a standalone RL algorithm. Many goal-conditioned methods learn a value function that captures the similarity between two states [29, 62, 91, 125]. Such distance functions are structurally similar to the critic function learned for contrastive learning, a connection we make precisely in Sec. 4. In fact, our analysis shows that C-learning [29] is already performing contrastive learning, and our experiments show that alternative contrastive RL methods can be much simpler and achieve higher performance.

Prior work has studied how representations related to reward functions using the framework of universal value functions [12, 106] and successor features [9, 52, 81]. While these methods typically require additional supervision to drive representation learning (manually-specified reward functions or features), our method is more similar to prior work that estimates the discounted state occupancy measure as an inner product between learned representations [11, 131]. While these methods use temporal difference learning, ours is akin to Monte Carlo learning. While Monte Carlo learning is often (but not always [23]) perceived as less sampling efficient, our experiments find that our approach can be as sample efficient as TD methods. Other prior work has focused on learning representations that can be used for planning [59, 82, 104, 105, 128]. Our method will learn representations using an objective similar to prior work [105, 109], but makes the key observation that the representation already encodes a value function: no additional planning or RL is necessary to choose actions.

Please see Appendix A for a discussion of how our work relates to unsupervised skill learning.

3 Preliminaries

Goal-conditioned reinforcement learning. The goal-conditioned RL problem is defined by states $s_t \in \mathcal{S}$, actions a_t, an initial state distribution $p_0(s)$, the dynamics $p(s_{t+1} \mid s_t, a_t)$, a distribution over goals $p_g(s_g)$, and a reward function $r_g(s, a)$ for each goal. This problem is equivalent to a multi-task RL [5, 45, 121, 130, 139], where tasks correspond to reaching goals states. Following prior work [11, 15, 29, 103], we define the reward as the probability (density) of reaching the goal at the next time step:[4]

$$r_g(s_t, a_t) \triangleq (1-\gamma) p(s_{t+1} = s_g \mid s_t, a_t). \tag{1}$$

This reward function is appealing because it avoids the need for a human user to specify a distance metric (unlike, e.g., [6]). Even though our method will not estimate the reward function, we will still use the reward function for analysis. For a goal-conditioned policy $\pi(a \mid s, s_g)$, we use $\pi(\tau \mid s_g)$ to denote the probability of sampling an infinite-length trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$. We defined the expected reward objective and Q-function as

$$\max_\pi \mathbb{E}_{p_g(s_g), \pi(\tau \mid s_g)} \left[\sum_{t=0}^{\infty} \gamma^t r_g(s_t, a_t) \right], \quad Q^\pi_{s_g}(s, a) \triangleq \mathbb{E}_{\pi(\tau \mid s_g)} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_g(s_{t'}, a_{t'}) \mid {}^{s_t = s,}_{a_t = a} \right]. \tag{2}$$

Intuitively, this objective corresponds to sampling a goal s_g and then optimizing the policy to go to that goal and stay there. Finally, we define the discounted state occupancy measure as [55, 142]

$$p^{\pi(\cdot \mid \cdot, s_g)}(s_{t+} = s) \triangleq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p_t^{\pi(\cdot \mid \cdot, s_g)}(s_t = s), \tag{3}$$

[4] At the initial state, this reward also includes the probability that the agent started at the goal: $r_g(s_0, a_0) = (1-\gamma)(p(s_1 = s_g \mid s_0, a_0) + p_0(s_0 = s_g))$

where $p_t^\pi(s)$ is the probability density over states that policy π visits after t steps. Sampling from the discounted state occupancy measure is easy: the first sample a time offset from a geometric distribution ($t \sim \textsc{Geom}(1-\gamma)$), and then look at what state the policy visits after exactly t steps. We will use s_{t+} to denote states sampled from the discounted state occupancy measure. Because our method will combine experience collected from multiple policies, we also define the average stationary distribution as $p^{\pi(\cdot|\cdot)}(s_{t+} = s \mid s, a) \triangleq \int p^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s \mid s, a) p^\pi(s_g \mid s, a) ds_g$, where $p^\pi(s_g \mid s, a)$ is the probability of the *commanded* goal given the current state-action pair. This stationary distribution is equivalent to that of the policy $\pi(a \mid s) \triangleq \int \pi(a \mid s, s_g) p^\pi(s_g \mid s) ds_g$ [145].

Contrastive representation learning. Contrastive representation learning methods [17, 46, 53, 54, 61, 77, 84, 86, 87, 122, 124, 129] take as input pairs of positive and negative examples, and learn representations so that positive pairs have similar representations and negative pairs have dissimilar representations. We use (u, v) to denote an input pair (e.g., u is an image, and v is an augmented version of that image). Positive examples are sampled from a joint distribution $p(u, v)$, while negative examples are sampled from the product of marginal distributions, $p(u)p(v)$. We will use an objective based on binary classification [77, 86, 87, 94]. Let $f(u, v) = \phi(u)^T \psi(v)$ be the similarity between the representations of u and v. We will call f the *critic function*[5] and note that its range is $(-\infty, \infty)$. We will use NCE-binary [84] objective (also known as InfoMAX [54]):

$$\max_{f(u,v)} \mathbb{E}_{\substack{(u,v^+)\sim p(u,v) \\ v^- \sim p(u)}} \Big[\log \sigma(\underbrace{f(u, v^+)}_{\phi(u)^T \psi(v^+)}) + \log(1 - \sigma(\underbrace{f(u, v^-)}_{\phi(u)^T \psi(v^-)})) \Big]. \tag{4}$$

4 Contrastive Learning as an RL Algorithm

This section shows how to use contrastive representation to *directly* perform goal-conditioned RL. The key idea (Lemma 4.1) is that contrastive learning estimates the Q-function for a certain policy and reward function. To prove this result, we relate the Q-function to the state occupancy measure (Sec. 4.1) and then relate the optimal critic function to the state occupancy measure (Sec. 4.2).

This result allows us to propose a new algorithm for goal-conditioned RL based on contrastive learning. Unlike prior work, this algorithm is not adding contrastive learning on top of an existing RL algorithm. This framework generalizes C-learning [29], offering a cogent explanation for its good performance while also suggesting new methods that are simpler and can achieve higher performance.

4.1 Relating the Q-function to probabilities

This section sets the stage for the main results of this section by providing a probabilistic perspective goal-conditioned RL. The expected reward objective and associated Q-function in (Eq. 2) can equivalently be expressed as the probability (density) of reaching a goal in the future:

Proposition 1 (rewards $\rightarrow$ probabilities)**.** *The Q-function for the goal-conditioned reward function r_g (Eq. 1) is equivalent to the probability of state s_g under the discounted state occupancy measure:*

$$Q^\pi_{s_g}(s, a) = p^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s_g \mid s, a). \tag{5}$$

The proof is in Appendix B. Translating rewards into probabilities not only makes it easier to analyze the goal-conditioned problem, but also means that any method for estimating probabilities (e.g., contrastive learning) can be turned into a method for estimating this Q-function.

4.2 Contrastive Learning Estimates a Q-Function

We will use contrastive learning to learn a value function by carefully choosing the inputs u and v. The first input, u, will correspond to a state-action pair, $u = (s_t, a_t) \sim p(s, a)$. In practice, these pairs are sampled from the replay buffer. Including the actions in the input is important because it will allow us to determine which actions to take to reach a desired future state. The second variable, v, is a *future* state, $v = s_f$. For the "positive" training pairs, the future state is sampled from the discounted

[5] In contrastive learning, the critic function indicates the similarity between a pair of inputs [99]; in RL, the critic function indicates the future expected returns [66]. Our method combines contrastive learning and RL in a way that these meanings become one and the same.

state occupancy measure, $s_f \sim p^{\pi(\cdot|\cdot)}(s_{t+} \mid s_t, a_t)$. For the "negative" training pairs, we sample a future state from a random state-action pair: $s_f \sim p(s_{t+}) \triangleq \int p^{\pi(\cdot|\cdot)}(s_{t+} \mid s, a)p(s, a)dsda$. With these inputs, the contrastive learning objective (Eq. 4) can be written as

$$\max_f \mathbb{E}_{\substack{(s,a)\sim p(s,a), s_f^- \sim p(s_f) \\ s_f^+ \sim p^{\pi(\cdot|\cdot)}(s_{t+}|s_t,a_t)}} \left[\mathcal{L}(s, a, s_f^+, s_f^-)\right],$$
$$\text{where} \quad \mathcal{L}(s, a, s_f^+, s_f^-) \triangleq \log \sigma(\underbrace{f(s, a, s_f^+)}_{\phi(s,a)^T\psi(s_f^+)}) + \log(1 - \sigma(\underbrace{f(s, a, s_f^-)}_{\phi(s,a)^T\psi(s_f^-)})). \tag{6}$$

Intuitively, the critic function $f(u = (s_t, a_t), v = s_f)$ now tells us the correlation between the current state-action pair and future outcomes, analogous to a Q-function. We therefore can use the critic function in the same way as actor-critic RL algorithms [66], figuring out which actions lead to the desired outcome. Because the Bayes-optimal critic function is a function of the state occupancy measure [84], $f^*(s, a, s_g) = \log\left(\frac{p^{\pi(\cdot|\cdot)}(s_{t+}=s_g|s,a)}{p(s_g)}\right)$, it can be used to express the Q-function:

Lemma 4.1. *The critic function that optimizes Eq. 6 is a Q-function for the goal-conditioned reward function (Eq. 1), up to a multiplicative constant* $\frac{1}{p(s_f)}$*:* $\exp(f^*(s, a, s_f)) = \frac{1}{p(s_f)} \cdot Q_{s_f}^{\pi(\cdot|\cdot)}(s, a)$.

The critic function can be viewed as an unnormalized density model, where $p(s_g)$ is the partition function. Much of the appeal of contrastive learning is it avoids estimating the partition function [46], which can be challenging; in the RL setting, it will turn out that this constant can be ignored when selecting actions. Our experiments show that learning a normalized density model works well when s_g is low-dimensional, but struggles to solve higher-dimensional tasks.

This lemma relates the critic function to $Q_{s_f}^{\pi(\cdot|\cdot)}(s, a)$, not $Q_{s_f}^{\pi(\cdot|\cdot, s_f)}(s, a)$. The underlying reason is that the critic function combines together experience collected when commanding different goals. Prior goal-conditioned behavioral cloning methods [22, 41, 83, 120] perform similar sharing, but do not analyze the relationship between the learned policies and Q functions. Sec. 4.5 shows that this critic function can be used as the basis for a convergent RL algorithm under some assumptions.

4.3 Learning the Goal-Conditioned Policy

The learned critic function not only tells us the likelihood of future states, but also tells us how different actions change the likelihood of a state occurring in the future. Thus, to learn a policy for reaching a goal state, we choose the actions that make that state most likely to occur in the future:

$$\max_{\pi(a|s,s_g)} \mathbb{E}_{\pi(a|s,s_g)p(s)p(s_g)}\left[f(s, a, s_f = s_g)\right] \approx \mathbb{E}_{\pi(a|s,s_g)p(s)p(s_g)}\left[\log Q_{s_g}^{\pi(\cdot|\cdot)}(s, a) - \log p(s_g)\right]. \tag{7}$$

The approximation above reflects errors in learning the optimal critic, and will allow us to prove that this policy loss corresponds to policy improvement in Sec. 4.5, under some assumptions.

In practice, we parametrize the goal-conditioned policy as a neural network that takes as input the state and goal and outputs a distribution over actions. The actor loss (Eq. 7) is computed by sampling states and *random* goals from the replay buffer, sampling actions from the policy, and then taking gradients on the policy using a reparametrization gradient. On tasks with image observations, we add an action entropy term to the policy objective.

4.4 A Complete Goal-Conditioned RL Algorithm

The complete algorithm alternates between fitting the critic function using contrastive learning, updating the policy using Eq. 7, and collecting more data. Alg. 1 provides a JAX [13] implementation of the actor and critic losses. Note that the critic is parameterized as an inner product between a representation of the state-action pair, and a representation of the goal state: $f(s, a, s_g) = \phi(s, a)^T\psi(s_g)$. This parameterization allows for efficient computation, as we can compute the goal representations just once, and use them both in the positive pairs and the negative pairs. While this is common practice in representation learning, it is not exploited by most goal-conditioned RL algorithms. We refer to this method as `contrastive RL (NCE)`. In Appendix C, we derive a variant of this method (`contrastive RL (CPC)`) that uses the infoNCE bound on mutual information.

Algorithm 1 Contrastive RL (NCE): the actor and critic losses for our method.

```
from jax.numpy import einsum, eye
from optax import sigmoid_binary_cross_entropy
def critic_loss(states, actions, future_states):
  sa_repr = sa_encoder(states, actions)  # (batch_dim, repr_dim)
  g_repr = g_encoder(future_states)      # (batch_dim, repr_dim)
  logits = einsum('ik,jk->ij', sa_repr, g_repr)  # <sa_repr[i], g_repr[j]> for all i,j
  return sigmoid_binary_cross_entropy(logits=logits, labels=eye(batch_size))

def actor_loss(states, goals):
  actions = policy.sample(states, goal=goals)  # (batch_size, action_dim)
  sa_repr = sa_encoder(states, actions)        # (batch_dim, repr_dim)
  g_repr = g_encoder(goals)                    # (batch_dim, repr_dim)
  logits = einsum('ik,ik->i', sa_repr, g_repr)  # <sa_repr[i], g_repr[i]>
  return -1.0 * logits
```

Contrastive RL (NCE) is an on-policy algorithm because it only estimates the Q-function for the policy that collected the data. However, in practice, we take as many gradient steps on each transition as standard off-policy RL algorithms [40, 48]. Please see Appendix E for full implementation details. We will also release an efficient implementation based on ACME [57] and JAX [13]. On a single TPUv2, training proceeds at $1100 \frac{\text{batches}}{\text{sec}}$ for state-based tasks and $105 \frac{\text{batches}}{\text{sec}}$ for image-based tasks; for comparison, our implementation of DrQ on the same hardware setup runs at $28 \frac{\text{batches}}{\text{sec}}$ ($3.9\times$ slower).[6] Architectures and hyperparameters are described in Appendix E.[7]

4.5 Convergence Guarantees

In general, providing convergence guarantees for methods that perform relabeling is challenging. Most prior work offers no guarantees [6, 22, 23] or guarantees under only restrictive assumptions [41, 120].

To prove that contrastive RL converges, we will introduce an additional filtering step into the method, throwing away some training examples. Precisely, we exclude training examples (s, a, s_f) if the probability of the corresponding trajectory $\tau_{i:j} = (s_i, a_i, s_{i+1}, a_{i+1}, \cdots, s_j, a_j)$ sampled from $\pi(\tau \mid s_g)$ under the commanded goal s_g is very different from the trajectory's probability under the actually-reached goal s_j:

$$\textsc{ExcludeTraj}(\tau_{i:j}) = \delta\left(\left|\frac{\pi(\tau_{i:j} \mid s_g)}{\pi(\tau_{i:j} \mid s_j)} - 1\right| > \epsilon\right).$$

While this modification is necessary to prove convergence, ablation experiments in Appendix Fig. 13 show that the filtering step can actually hurt performance in practice, so we do not include this filtering step in the experiments in the main text. We can now prove that contrastive RL performs approximate policy improvement.

Lemma 4.2 (Approximate policy improvement)**.** *Assume that states and actions are tabular and assume that the critic is Bayes-optimal. Let $\pi'(a \mid s, s_g)$ be the goal-conditioned policy obtained after one iteration of contrastive RL with a filtering parameter of ϵ. Then this policy achieves higher rewards than the initial goal-conditioned policy:*

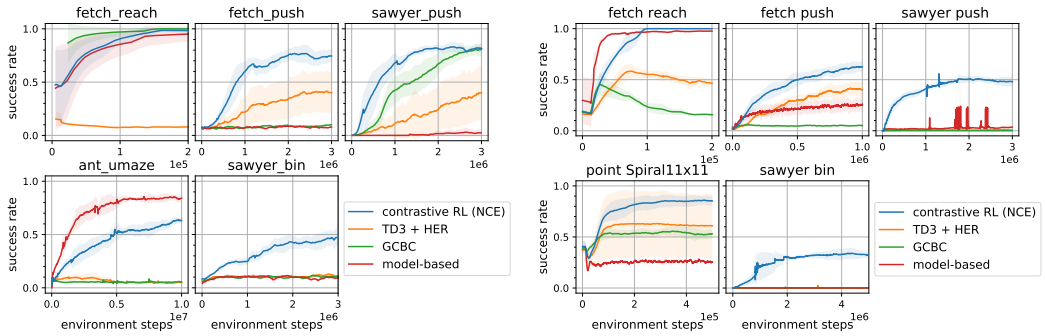
$$\mathbb{E}_{\pi'(\tau|s_g)}\left[\sum_{t=0}^{\infty} \gamma^t r_{s_g}(s_t, a_t)\right] \geq \mathbb{E}_{\pi(\tau|s_g)}\left[\sum_{t=0}^{\infty} \gamma^t r_{s_g}(s_t, a_t)\right] - \frac{2\gamma\epsilon}{1-\gamma} \qquad \textit{for all goals } s_g \in \{s_g \mid p_g(s_g) > 0\}.$$

The proof is in Appendix B. This result shows that performing contrastive RL on static dataset results in one step of approximate policy improvement. Re-collecting data and then applying contrastive RL over and over again corresponds to approximate policy improvement (see [10, Lemma 6.2]).

In summary, we have shown that applying contrastive learning to a particular choice of inputs results in an RL algorithm, one that learns a Q-function and (under some assumptions) converges to the reward-maximizing policy. Contrastive RL (NCE) is simple: it does not require multiple Q-values [40], target Q networks [88], data augmentation [69, 73], or auxiliary objectives [116, 137].

[6] The more recent DrQ-v2 [136] uses on 1 NVIDIA V100 GPU to achieve a training speed of $96/4 = 24 \frac{\text{batches}}{\text{sec}}$. The factor of 4 comes from an action repeat of 2 and an update interval of 2.

[7] Code and more results are available: `https://ben-eysenbach.github.io/contrastive_rl`



(a) State-based tasks

(b) Image-based tasks

Figure 2: **Goal-conditioned RL.** Contrastive RL (NCE) outperforms prior methods on most tasks. **Baselines**: `HER` [80] is a prototypical actor-critic method that uses hindsight relabeling [6]; `Goal-conditioned behavioral cloning (GCBC)` [22, 41, 83, 117] performs behavior cloning on relabeled experience; `model-based` fits a density model to the discounted state occupancy measure, similar on [21, 23, 60].

4.6 C-learning as Contrastive Learning

C-learning [29] is a special case of contrastive RL: it learns a critic function to distinguish future goals from random goals. Compared with contrastive RL (NCE), C-learning learns the classifier using temporal difference learning.[8] Viewing C-learning as a special case of contrastive RL suggests that contrastive RL algorithms might be implemented in a variety of different ways, each with relative merits. For example, `contrastive RL (NCE)` is much simpler than C-learning and tends to perform a bit better. Appendix D introduces another member of the contrastive RL family (`contrastive RL (NCE + C-learning)`) that tends to yield the best performance .

5 Experiments

Our experiments use goal-conditioned RL problems to compare contrastive RL algorithms to prior non-contrastive methods, including those that use data augmentation and auxiliary objectives. We then compare different members of the contrastive RL family, and show how contrastive RL can be effectively applied to the offline RL setting. Appendices E, F, and G contain experiments, visualizations, and failed experiments.

5.1 Comparing to prior goal-conditioned RL methods

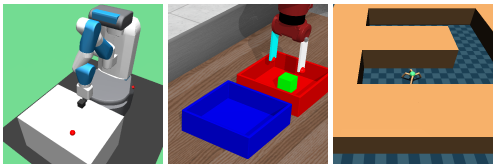


Figure 3: **Environments.** We show a subset of the goal-conditioned environments used in our experiments.

Baselines. We compare three baselines. "HER" [80] is a goal-conditioned RL method that uses hindsight relabeling [6] with a high-performance actor-critic algorithm (TD3). This baseline is representative of a large class of prior work that uses hindsight relabeling [6, 76, 102, 106]. Like contrastive RL, this baseline does not assume access to a reward function. The second baseline is goal-conditioned behavioral cloning ("GCBC") [16, 22, 25, 41, 83, 96, 117, 120], which trains a policy to reach goal s_g by performing behavioral cloning on trajectories that reach state s_g. GCBC is a simple method that achieves excellent results [16, 25] and has the same inputs as our method ((s, a, s_f) triplets). A third baseline is a model-based approach that fits a density model to the future state distribution $p^{\pi(\cdot|\cdot)}(s_{t+} \mid s, a)$ and trains a goal-conditioned policy to maximize the probability of the commanded goal. This baseline is similar to successor representations [21] and prior multi-step models [23, 60]. Both contrastive RL (Alg. 1) and this model-based approach encode the future state distribution, but the output dimension of this model-based method depends on the state dimension. We, therefore, expect this approach to excel in low-dimensional settings but struggle with image-based tasks. Where possible, we use the same hyperparameters for all methods. We will

[8] The objectives are subtly different: C-learning estimates the probability that policy $\pi(\cdot \mid \cdot, s_g)$ visits state $s_f = s_g$, whereas contrastive RL (NCE) estimates the probability that any of the goal conditioned policies visit state s_f.

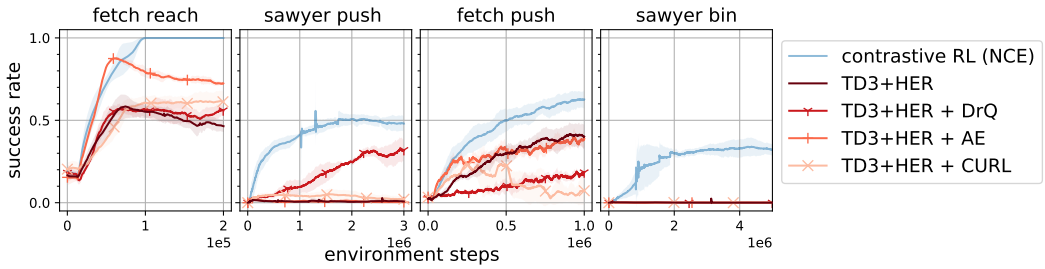


Figure 4: **Representation learning for image-based tasks.** While adding data augmentation and auxiliary representation objectives can boost the performance of the `TD3+HER` baseline, replacing the underlying goal-conditioned RL algorithm with one that resembles contrastive representation learning (i.e., ours) yields a larger increase in success rates. **Baselines**: `DrQ` [69] augments images and averages the Q-values across 4 augmentations; `auto encoder (AE)` adds an auxiliary reconstruction loss [32, 91, 93, 137]; `CURL` [116] applies RL on top of representations learned via augmentation-based contrastive learning.

include additional representation learning baselines when studying representations in the subsequent section.

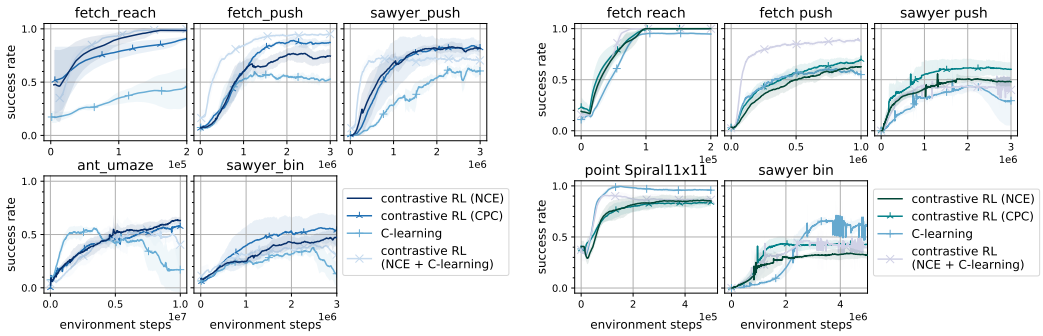
Tasks. We compare it to a suite of goal-conditioned tasks, mostly taken from prior work. Four standard manipulation tasks include `fetch reach` and `fetch push` from Plappert et al. [97] and `sawyer push` and `sawyer bin` from Yu et al. [139]. We evaluate these tasks both with state-based observations and (unlike most prior work) image-based observations. The `sawyer bin` task poses an exploration challenge, as the agent must learn to pick up an object from one bin and place it at a goal location in another bin; the agent does not receive any reward shaping or demonstrations. We include two navigation tasks: `point Spiral11x11` is a 2D maze task with image observations and `ant umaze` [36] is a 111-dimensional locomotion task that presents a challenging low-level control problem. Where possible, we use the same initial state distribution, goal distribution, observations, and definition of success as prior work. Goals have the same dimension as the states, with one exception: on the `ant umaze` task, we used the global XY position as the goal. We illustrate three of the tasks to the right. The agent does not have access to any ground truth reward function.

We report results in Fig. 2, using five random seeds for each experiment and plotting the mean and standard deviation across those random seeds. On the state-based tasks (Fig. 2a), most methods solve the easiest task (`fetch reach`) while only our method solves the most challenging task (`sawyer bin`). Our method also outperforms all prior methods on the two pushing tasks. The model-based baseline performs best on the `ant umaze` task, likely because learning a model is relatively easy when the goal is lower-dimensional (just the XY location). On the image-based tasks (Fig. 2b), most methods make progress on the two easiest tasks (`fetch reach` and `point Spiral11x11`); our method outperforms the baselines on the three more challenging tasks. Of particular note is the success on `sawyer push` and `sawyer bin`: while the success rate of our method remains below 50%, no baselines make any progress on learning these tasks. These results suggest that contrastive RL (NCE) is a competitive goal-conditioned RL algorithm.

5.2 Comparing to prior representation learning methods

We hypothesize that contrastive RL may automatically learn good representations. To test this hypothesis, we compare contrastive RL (NCE) to techniques proposed by prior work for representation learning. These include data augmentation [69, 73, 136] ("DrQ") and auxiliary objectives based on an autoencoder [32, 91, 93, 137] ("AE") and a contrastive learning objective ("CURL") that generates positive examples using data augmentation, similar to prior work [89, 116, 118]. Because prior work has demonstrated these techniques in combination with actor-critic RL algorithms, we will use these techniques in combination with the actor-critic baseline from the previous section ("TD3 + HER"). While contrastive RL (NCE) resembles a contrastive representation learning method, it does not include any data augmentation or auxiliary representation learning objectives.

We show results in Fig. 4, with error bars again showing the mean and standard deviation across 5 random seeds. While adding the autoencoder improves the baseline on the `fetch reach` and adding DrQ improves the baseline on the `sawyer push`, contrastive RL (NCE) outperforms the prior methods on all tasks. Unlike these methods, contrastive RL does not use auxiliary objectives or additional domain knowledge in the form of image-appropriate data augmentations. These experiments do not show that representation learning is never useful, and do not show that contrastive



(a) state-based observations (b) image-based observations

Figure 5: **Contrastive RL design decisions.** Generalizing C-learning to a family of contrastive RL algorithms allowed us to identify algorithms that are much simpler (contrastive RL (NCE)) and that consistently achieve higher performance (contrastive RL (NCE + C-learning)).

RL cannot be improved with additional representation learning machinery. Rather, they show that designing RL algorithms that structurally resemble contrastive representation learning yields bigger improvements than simply adding representation learning tricks on top of existing RL algorithms.

5.3 Probing the dimensions of contrastive RL

Up to now, we have focused on the specific instantiation of contrastive RL spelled out in Alg. 1. However, there is a whole family of RL algorithms with contrastive characteristics. C-learning is a contrastive RL algorithm that uses temporal difference learning (Sec. 4.6). Contrastive RL (CPC) is a variant of Alg. 1 based on the infoNCE objective [95] that we derive in Appendix C Contrastive RL (NCE + C-learning) is a variant that combines C-learning with Alg. D (see Appendix D.). The aim of these experiments are to study whether generalizing C-learning to a family of contrastive RL algorithms was useful: do the simpler methods achieve similar performance, and do other methods achieve better performance?

We present results in Fig. 5, again plotting the mean and standard deviation across five random seeds. Contrastive RL (CPC) outperforms contrastive RL (NCE) on three, suggesting that swapping one mutual information estimator for another can sometimes improve performance, though both estimators can be effective. C-learning outperforms contrastive RL (NCE) on three tasks but performs worse on other tasks. Contrastive RL (NCE + C-learning) consistently ranks among the best methods. These experiments demonstrate that the prior contrastive RL method, C-learning [29], achieves good results on most tasks; generalizing C-learning to a family of contrastive RL algorithms resulting in new algorithms that achieve higher performance and can be much simpler.

5.4 Partial Observability and Moving Cameras

Many realistic robotics tasks exhibit partial observability, and have cameras that are not fixed but rather attached to moving robot parts. Our next experiment tests if contrastive RL can cope with these sorts of challenges. To study this question, we modified the `sawyer push` task so that the camera tracks the hand at a fixed distance, as if it were rigidly mounted to the arm. This means that, at the start of the episode, the scene is occluded by the wall at the edge of the table, so the agent cannot see the location of the puck (see Fig. 6 (left)). Nonetheless, contrastive RL (NCE) successfully handles this partial observability, achieving a success rate of around 35%. Fig. 6 (left) shows an example rollout and Fig. 6 (right) shows the learning curve. For comparison, the success rate when using the fixed static camera was 75%. Taken together, these results suggest that contrastive RL can cope with moving cameras and partial observability, while also suggesting that improved strategies (e.g., non-Markovian architectures) might achieve even better results.

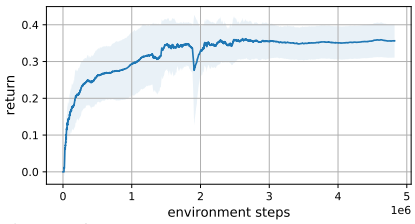


Figure 6: **Partial observability and moving cameras.** Contrastive RL can solve partially observed tasks.

Table 1: **Offline RL on D4RL AntMaze** [36]. Contrastive RL outperforms all baselines in 5 out of 6 tasks.

	no TD					uses TD	
	BC	DT	GCBC	Contrastive RL + BC 2 nets	Contrastive RL + BC 5 nets	TD3+BC*	IQL*
umaze-v2	54.6	65.6	65.4	81.9 (±1.7)	79.8 (±1.4)	78.6	**87.5**
umaze-diverse-v2	45.6	51.2	60.9	**75.4** (±3.5)	**77.6** (±2.8)	71.4	62.2
medium-play-v2	0.0	1.0	58.1	**71.5** (±5.2)	**72.6** (±2.9)	10.6	**71.2**
medium-diverse-v2	0.0	0.6	67.3	**72.5** (±2.8)	**71.5** (±1.3)	3.0	**70.0**
large-play-v2	0.0	0.0	32.4	41.6 (±6.0)	**48.6** (±4.4)	0.2	39.6
large-diverse-v2	0.0	0.2	36.9	**49.3** (±6.3)	**54.1** (±5.5)	0.0	47.5

* While TD3+BC and IQL report results on the `-v0` tasks, the change to `-v2` has a negligible effect on TD methods [8].

5.5 Contrastive RL for Offline RL

Our final experiment studies whether the benefits from contrastive RL (NCE) transfer to the offline RL setting, where the agent is prohibited from interacting with the environment. We use the benchmark AntMaze tasks from the D4RL benchmark [36], as these are goal-conditioned tasks commonly studied in the offline setting.

We adapt contrastive RL (NCE) to the offline setting by adding an additional (goal-conditioned) behavioral cloning term to the policy objective (Eq. 7), using a coefficient of λ:

$$\max_{\pi(a|s,s_g)} \mathbb{E}_{\pi(a|s,s_g)p(s,a_{\text{orig}},s_g)} \left[(1-\lambda) \cdot f(s,a,s_f = s_g) + \lambda \cdot \log \pi(a_{\text{orig}} \mid s, s_g)\right].$$

Note that setting $\lambda = 1$ corresponds to GCBC [16, 22, 25, 41, 83, 96, 117, 120], which we will include as a baseline. Following TD3+BC [39], we learn multiple critic functions (2 and 5) and take the minimum when computing the actor update. We also compare to prior offline RL methods that eschew TD learning: (unconditional) behavioral cloning (BC), the implementation of GCBC from [25] (which refers to GCBC as RvS-G), and a recent method based on the transformer architecture (DT [16]). Lastly, we compare with two more complex methods that use TD learning: TD3+BC [39] and IQL [68]. Unlike contrastive RL and GCBC, these TD learning methods do not perform goal relabeling. We use the numbers reported for these baselines in prior work [25, 68].

As shown in Table 1, contrastive RL (NCE) outperforms all baselines on five of the six benchmark tasks. Of particular note are the most challenging "-large" tasks, where contrastive RL achieves a 7% to 9% absolute improvement over IQL. We note that IQL does not use goal relabeling, which is the bedrock of contrastive RL. Compared to baselines that do not use TD learning, the benefits are more pronounced, with a median (absolute) improvement over GCBC of 15%. The performance of contrastive RL improves when increasing the number of critics from 2 to 5, suggesting that the key to solving more challenging offline RL tasks may be increased capacity, rather than TD learning. Taken together, these results show the value of contrastive RL for offline goal-conditioned tasks.

6 Conclusion

In this paper, we showed how contrastive representation learning can be used for goal-conditioned RL. This connection not only lets us re-interpret a prior RL method as performing contrastive learning, but also suggests a family of contrastive RL methods, which includes simpler algorithms, as well as algorithms that attain better overall performance. While this paper might be construed to imply that RL is more or less important than representation learning [72, 75, 112, 114], we have a different takeaway: that it may be enough to build RL algorithms that *look like* representation learning.

One limitation of this work is that it looks only at the goal-conditioned RL problems. How these methods might be applied to arbitrary RL problems remains an open problem, though we note that recent algorithms for this setting [28] already bear a resemblance to contrastive RL. Whether the rich set of ideas from contrastive learning might be used to construct even better RL algorithms likewise remains an open question.

Acknowledgements. Thanks to Hubert Tsai, Martin Ma, and Simon Kornblith for discussions about contrastive learning. Thanks to Kamyar Ghasemipour, Suraj Nair, and anonymous reviewers for feedback on the paper. Thanks to Ofir Nachum, Daniel Zheng, and the JAX and Acme teams for helping to release and debug the code. This material is supported by the Fannie and John Hertz Foundation and the NSF GRFP (DGE1745016). UC Berkeley research is also supported by gifts from Alibaba, Amazon Web Services, Ant Financial, CapitalOne, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, Nvidia, Scotiabank, Splunk and VMware.

References

[1] Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. (2018). Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*.

[2] Achiam, J., Knight, E., and Abbeel, P. (2019). Towards characterizing divergence in deep Q-learning. *arXiv preprint arXiv:1903.08894*.

[3] Alain, G. and Bengio, Y. (2016). Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

[4] Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M.-A., and Hjelm, R. D. (2019). Unsupervised state representation learning in Atari. *Advances in Neural Information Processing Systems*, 32.

[5] Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR.

[6] Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. In *NeurIPS*.

[7] Annasamy, R. M. and Sycara, K. (2019). Towards better interpretability in deep Q-networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4561–4569.

[8] Authors, I. (2022). Private Communication.

[9] Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30.

[10] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

[11] Blier, L., Tallec, C., and Ollivier, Y. (2021). Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*.

[12] Borsa, D., Barreto, A., Quan, J., Mankowitz, D., Munos, R., Van Hasselt, H., Silver, D., and Schaul, T. (2018). Universal successor features approximators. *arXiv preprint arXiv:1812.07626*.

[13] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

[14] Brown, D., Goo, W., Nagarajan, P., and Niekum, S. (2019). Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR.

[15] Chane-Sane, E., Schmid, C., and Laptev, I. (2021). Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pages 1430–1440. PMLR.

[16] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34.

[17] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. (2020). A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709.

[18] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15745–15753.

[19] Choi, J., Sharma, A., Lee, H., Levine, S., and Gu, S. S. (2021). Variational empowerment as representation learning for goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pages 1953–1963. PMLR.

[20] Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*.

[21] Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624.

[22] Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. (2019). Goal-conditioned imitation learning. *Advances in Neural Information Processing Systems*, 32:15324–15335.

[23] Dosovitskiy, A. and Koltun, V. (2016). Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*.

[24] Du, Y., Gan, C., and Isola, P. (2021). Curious representation learning for embodied intelligence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10408–10417.

[25] Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. (2021). Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*.

[26] Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. (2020). Rewriting history with inverse RL: Hindsight inference for policy improvement. *ArXiv*, abs/2002.11089.

[27] Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*.

[28] Eysenbach, B., Levine, S., and Salakhutdinov, R. R. (2021a). Replacing rewards with examples: Example-based policy search via recursive classification. *Advances in Neural Information Processing Systems*, 34.

[29] Eysenbach, B., Salakhutdinov, R., and Levine, S. (2021b). C-learning: Learning to achieve goals via recursive classification. *ArXiv*, abs/2011.08909.

[30] Eysenbach, B., Salakhutdinov, R. R., and Levine, S. (2019). Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

[31] Eysenbach, B., Udatha, S., Levine, S., and Salakhutdinov, R. (2022). Imitating past successes can be very suboptimal. *arXiv preprint arXiv:2206.03378*.

[32] Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE.

[33] Fischinger, D., Vincze, M., and Jiang, Y. (2013). Learning grasps for unknown objects in cluttered scenes. In *2013 IEEE international conference on robotics and automation*, pages 609–616. IEEE.

[34] Florensa, C., Degrave, J., Heess, N., Springenberg, J. T., and Riedmiller, M. (2019). Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*.

[35] Florensa, C., Held, D., Geng, X., and Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR.

[36] Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2020). D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*.

[37] Fu, J., Luo, K., and Levine, S. (2017). Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.

[38] Fu, J., Singh, A., Ghosh, D., Yang, L., and Levine, S. (2018). Variational inverse control with events: A general framework for data-driven reward definition. In *NeurIPS*.

[39] Fujimoto, S. and Gu, S. S. (2021). A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34.

[40] Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR.

[41] Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C. M., Eysenbach, B., and Levine, S. (2020). Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*.

[42] Gregor, K., Rezende, D. J., and Wierstra, D. (2016). Variational intrinsic control. *arXiv preprint arXiv:1611.07507*.

[43] Grill, J.-B., Strub, F., Altch'e, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733.

[44] Guo, Z. D., Azar, M. G., Piot, B., Pires, B. A., and Munos, R. (2018). Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*.

[45] Guo, Z. D., Pires, B. A., Piot, B., Grill, J.-B., Altché, F., Munos, R., and Azar, M. G. (2020). Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pages 3875–3886. PMLR.

[46] Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2).

[47] Ha, D. and Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

[48] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

[49] Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019a). Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

[50] Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019b). Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR.

[51] Han, T., Xie, W., and Zisserman, A. (2020). Self-supervised co-training for video representation learning. *Advances in Neural Information Processing Systems*, 33:5679–5690.

[52] Hansen, S., Dabney, W., Barreto, A., Van de Wiele, T., Warde-Farley, D., and Mnih, V. (2019). Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*.

[53] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

[54] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

[55] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573.

[56] Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer.

[57] Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K., Henderson, S., Novikov, A., Colmenarejo, S. G., Cabi, S., Gulcehre, C., Paine, T. L., Cowie, A., Wang, Z., Piot, B., and de Freitas, N. (2020). Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*.

[58] Hong, Z.-W., Yang, G., and Agrawal, P. (2022). Bilinear value networks. *arXiv preprint arXiv:2204.13695*.

[59] Ichter, B., Sermanet, P., and Lynch, C. (2020). Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv preprint arXiv:2010.06491*.

[60] Janner, M., Mordatch, I., and Levine, S. (2020). gamma-models: Generative temporal difference learning for infinite-horizon prediction. *Advances in Neural Information Processing Systems*, 33:1724–1735.

[61] Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

[62] Kaelbling, L. P. (1993). Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer.

[63] Kalashnikov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. (2021). Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv*, abs/2104.08212.

[64] Kish, L. (1965). Survey sampling. *John Wiley & Sons*.

[65] Klingemann, M. (2016). Raster fairy. `https://github.com/bmcfee/RasterFairy`.

[66] Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, 12.

[67] Konyushkova, K., Zolna, K., Aytar, Y., Novikov, A., Reed, S., Cabi, S., and de Freitas, N. (2020). Semi-supervised reward learning for offline reinforcement learning. *arXiv preprint arXiv:2012.06899*.

[68] Kostrikov, I., Nair, A., and Levine, S. (2021). Offline reinforcement learning with implicit Q-learning. *arXiv preprint arXiv:2110.06169*.

[69] Kostrikov, I., Yarats, D., and Fergus, R. (2020). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*.

[70] Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. (2020). Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*.

[71] Lange, S. and Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[72] Langford, J. (2010). Specializations of the master problem.

[73] Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. (2020). Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895.

[74] Laskin, M., Liu, H., Peng, X. B., Yarats, D., Rajeswaran, A., and Abbeel, P. (2021). CIC: Contrastive intrinsic control for unsupervised skill discovery. In *Deep RL Workshop NeurIPS 2021*.

[75] LeCun, Y. (2016). Predictive learning. `https://www.youtube.com/watch?v=0unt2Y4qxQo`. Keynote Talk.

[76] Levy, A., Konidaris, G., Platt, R., and Saenko, K. (2017). Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*.

[77] Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27.

[78] Li, A., Pinto, L., and Abbeel, P. (2020). Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767.

[79] Liang, Y., Machado, M. C., Talvitie, E., and Bowling, M. (2015). State of the art control of Atari games using shallow reinforcement learning. *arXiv preprint arXiv:1512.01563*.

[80] Lin, X., Baweja, H. S., and Held, D. (2019). Reinforcement learning without ground-truth state. *ArXiv*, abs/1905.07866.

[81] Liu, H. and Abbeel, P. (2021). Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR.

[82] Liu, K., Kurutach, T., Tung, C., Abbeel, P., and Tamar, A. (2020). Hallucinative topological memory for zero-shot visual planning. In *International Conference on Machine Learning*, pages 6259–6270. PMLR.

[83] Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. (2020). Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR.

[84] Ma, Z. and Collins, M. (2018). Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *EMNLP*.

[85] Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., and Pathak, D. (2021). Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34.

[86] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

[87] Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *ICML*.

[88] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[89] Nachum, O., Gu, S., Lee, H., and Levine, S. (2018a). Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*.

[90] Nachum, O., Gu, S. S., Lee, H., and Levine, S. (2018b). Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31.

[91] Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). Visual reinforcement learning with imagined goals. *Advances in Neural Information Processing Systems*, 31:9191–9200.

[92] Nair, S., Mitchell, E., Chen, K., Savarese, S., Finn, C., et al. (2022). Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR.

[93] Nasiriany, S., Pong, V. H., Lin, S., and Levine, S. (2019). Planning with goal-conditioned policies. In *NeurIPS*.

[94] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29.

[95] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[96] Paster, K., McIlraith, S. A., and Ba, J. (2020). Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419*.

[97] Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

[98] Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. (2019). Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*.

[99] Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., and Tucker, G. (2019). On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR.

[100] Qiu, S., Wang, L., Bai, C., Yang, Z., and Wang, Z. (2022). Contrastive ucb: Provably efficient contrastive self-supervised learning in online reinforcement learning. In *International Conference on Machine Learning*, pages 18168–18210. PMLR.

[101] Rakelly, K., Gupta, A., Florensa, C., and Levine, S. (2021). Which mutual-information representation learning objectives are sufficient for control? *ArXiv*, abs/2106.07278.

[102] Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. (2018). Learning by playing solving sparse reward tasks from scratch. In *International conference on machine learning*, pages 4344–4353. PMLR.

[103] Rudner, T. G., Pong, V., McAllister, R., Gal, Y., and Levine, S. (2021). Outcome-driven reinforcement learning via variational inference. *Advances in Neural Information Processing Systems*, 34.

[104] Rybkin, O., Zhu, C., Nagabandi, A., Daniilidis, K., Mordatch, I., and Levine, S. (2021). Model-based reinforcement learning via latent-space collocation. In *International Conference on Machine Learning*, pages 9190–9201. PMLR.

[105] Savinov, N., Dosovitskiy, A., and Koltun, V. (2018). Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*.

[106] Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR.

[107] Schmeckpeper, K., Xie, A., Rybkin, O., Tian, S., Daniilidis, K., Levine, S., and Finn, C. (2020). Learning predictive models from observation and interaction. In *European Conference on Computer Vision*, pages 708–725. Springer.

[108] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823.

[109] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. (2018). Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE.

[110] Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. (2019). Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*.

[111] Shu, R., Nguyen, T., Chow, Y., Pham, T., Than, K., Ghavamzadeh, M., Ermon, S., and Bui, H. (2020). Predictive coding for locally-linear control. In *International Conference on Machine Learning*, pages 8862–8871. PMLR.

[112] Silver, D., Singh, S., Precup, D., and Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, 299:103535.

[113] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*.

[114] Srinivas, A. and Abbeel, P. (2021). Unsupervised learning for reinforcement learning. Tutorial.

[115] Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. (2018). Universal planning networks. *ArXiv*, abs/1804.00645.

[116] Srinivas, A., Laskin, M., and Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*.

[117] Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. (2019). Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*.

[118] Stooke, A., Lee, K., Abbeel, P., and Laskin, M. (2021). Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR.

[119] Such, F. P., Madhavan, V., Liu, R., Wang, R., Castro, P. S., Li, Y., Zhi, J., Schubert, L., Bellemare, M. G., Clune, J., et al. (2018). An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. *arXiv preprint arXiv:1812.07069*.

[120] Sun, H., Li, Z., Liu, X., Zhou, B., and Lin, D. (2019). Policy continuation with hindsight inverse dynamics. *Advances in Neural Information Processing Systems*, 32:10265–10275.

[121] Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30.

[122] Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer.

[123] Tsai, Y.-H., Zhao, H., Yamada, M., Morency, L.-P., and Salakhutdinov, R. (2020). Neural methods for point-wise dependency estimation. In *Proceedings of the Neural Information Processing Systems Conference (Neurips)*.

[124] Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. (2019). On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*.

[125] Venkattaramanujam, S., Crawford, E., Doan, T. V., and Precup, D. (2019). Self-supervised learning of distance functions for goal-conditioned reinforcement learning. *ArXiv*, abs/1907.02998.

[126] Wang, H., Miahi, E., White, M., Machado, M. C., Abbas, Z., Kumaraswamy, R., Liu, V., and White, A. (2022). Investigating the properties of neural network representations in reinforcement learning. *arXiv preprint arXiv:2203.15955*.

[127] Warde-Farley, D., Van de Wiele, T., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. (2018). Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*.

[128] Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28.

[129] Weinberger, K. Q. and Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In *NIPS*.

[130] Wilson, A., Fern, A., Ray, S., and Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022.

[131] Wu, Y., Tucker, G., and Nachum, O. (2018a). The Laplacian in RL: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*.

[132] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018b). Unsupervised feature learning via non-parametric instance discrimination. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

[133] Xie, A., Singh, A., Levine, S., and Finn, C. (2018). Few-shot goal inference for visuomotor learning and planning. In *Conference on Robot Learning*, pages 40–52. PMLR.

[134] Xu, D. and Denil, M. (2019). Positive-unlabeled reward learning. *arXiv preprint arXiv:1911.00459*.

[135] Yang, G., Ajay, A., and Agrawal, P. (2021). Overcoming the spectral bias of neural value approximation. In *International Conference on Learning Representations*.

[136] Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. (2021a). Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*.

[137] Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. (2021b). Improving sample efficiency in model-free reinforcement learning from images. In *AAAI*.

[138] Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020a). Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

[139] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2020b). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR.

[140] Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. (2020a). Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*.

[141] Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., and Levine, S. (2019). Solar: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 7444–7453. PMLR.

[142] Zhang, S., Liu, B., and Whiteson, S. (2020b). Gradientdice: Rethinking generalized offline estimation of stationary values. In *International Conference on Machine Learning*, pages 11194–11203. PMLR.

[143] Zhang, T., Ren, T., Yang, M., Gonzalez, J., Schuurmans, D., and Dai, B. (2022). Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning*, pages 26447–26466. PMLR.

[144] Zhao, R., Sun, X., and Tresp, V. (2019). Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7553–7562. PMLR.

[145] Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

[146] Zolna, K., Reed, S., Novikov, A., Colmenarejo, S. G., Budden, D., Cabi, S., Denil, M., de Freitas, N., and Wang, Z. (2019). Task-relevant adversarial imitation learning. *arXiv preprint arXiv:1910.01077*.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The main claims are that (1) contrastive learning can be used to learn a Q-function (Proof in Appendix B) and that (2) contrastive RL methods can outperform non-contrastive RL algorithms on goal-conditioned RL tasks (results in Fig. 2).
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] While RL broadly might be used for applications with both positive and negative outcomes, our algorithmic contributions are not tied to any particular application.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We have included all experimental details in Appendix E; code will be released upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All figures show 5 random seeds, witht error bars corresponding to the mean and standard deviation across these seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Sec. 4.4 describes the training speed on one TPUv2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Related Work

Our work is also related to unsupervised skill discovery [1, 19, 27, 42, 52, 74, 110], in that the algorithm learns multiple policies by interacting in the environment without a reward function. Both these skill learning algorithms and our contrastive algorithm optimize a lower bound on mutual information. Indeed, prior work has discussed the close connection between mutual information and goal-conditioned RL [19, 127]. The key challenge in making this connection is *grounding* the skills, so that each skill corresponds to a specific goal-conditioned policy. While the skills can be grounded by manually-specifying the critic used for maximizing mutual information [19], manually-specifying the critic for high-dimensional tasks (e.g., images) would be challenging. Our work takes a different approach to grounding, one based on reasoning directly about continuous probabilities. In the end, our method will learn skills that each corresponds to a specific goal-conditioned policy and will be scalable to high-dimensional tasks.

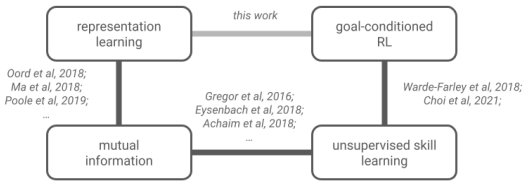


Figure 7: **Connecting related work.** This work helps draw connections between prior work, filling in a missing link.

Fig. 7 highlights some of the connections between related work. Prior work has thoroughly explained how many representation learning methods correspond to a lower bound on mutual information [84, 99]. Prior work in RL has proposed unsupervised skill learning algorithms using similar mutual information objectives [1, 27, 42], and more recent work has connected these unsupervised skills learning algorithms to goal-reaching. The key contribution of this paper is to connect representation learning to goal-conditioned RL.

B Proofs

B.1 Q-function are equivalent to the discounted state occupancy measure

This section proves Proposition 1. We start by recalling the definition of the discounted state occupancy measure (Eq. 3):

$$p(s_{t+} = s_g) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p_t^{\pi(\cdot|\cdot,s_g)}(s_t = s_g). \tag{8}$$

We first analyze the term for $t = 0$, and then analyze the term for $t > 0$. The probability of visiting a state at time $t = 0$ is just the initial state distribution:

$$p_0^{\pi(\cdot|\cdot,s_g)}(s_t = s_g) = p_0(s_0 = s_g).$$

We can now rewrite Eq. 8 as

$$p(s_{t+} = s_g) = (1-\gamma) p_0(s_0 = s_g) + (1-\gamma) \sum_{t=1}^{\infty} \gamma^t p_t^{\pi(\cdot|\cdot,s_g)}(s_t = s_g). \tag{9}$$

For $t > 1$, we can write the term as follows:

$$\begin{aligned} p_t^{\pi(\cdot|\cdot,s_g)}(s_t = s_g) &= \mathbb{E}_{p_{t-1}^{\pi(\cdot|\cdot,s_g)}(s_{t-1})\pi(a_{t-1}|s_{t-1},s_g)} \left[p_t(s_t = s_g \mid s_{t-1}, a_{t-1}) \right] \\ &= \mathbb{E}_{p_{t-1}^{\pi(\cdot|\cdot,s_g)}(s_{t-1}),\pi(a_{t-1}|s_{t-1},s_g)} \left[p(s_t = s_g \mid s_{t-1}, a_{t-1}) \right] \\ &= \mathbb{E}_{\tau \sim \pi(\tau|s_t)} \left[p(s_t = s_g \mid s_{t-1}, a_{t-1}) \right]. \end{aligned}$$

In the second line, we have used the Markov property to say that the probability of visiting s_g at time t depends only on dynamics, $p(s_{t+1} \mid s_t, a_t)$. In the third line, we have rewritten the expectation over trajectories, using s_{t-1} and a_{t-1} and the $t-1^{\text{th}}$ state-action pair in the trajectory. Substituting this

into Eq. 9, we get

$$
\begin{aligned}
p(s_{t+} = s_g) &= (1-\gamma)p_0(s_0 = s_g) + (1-\gamma)\sum_{t=1}^{\infty}\gamma^t \mathbb{E}_{\tau\sim\pi(\tau|s_g)}\left[p(s_t = s_g \mid s_{t-1}, a_{t-1})\right] \\
&= (1-\gamma)p_0(s_0 = s_g) + (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{\tau\sim\pi(\tau|s_g)}\left[p(s_{t+1} = s_g \mid s_t, a_t)\right] \\
&= (1-\gamma)p_0(s_0 = s_g) + (1-\gamma)\mathbb{E}_{\tau\sim\pi(\tau|s_g)}\left[\sum_{t=0}^{\infty}\gamma^t p(s_{t+1} = s_g \mid s_t, a_t)\right] \\
&= \mathbb{E}_{\tau\sim\pi(\tau|s_g)}\left[(1-\gamma)p_0(s_0 = s_g) + (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p(s_{t+1} = s_g \mid s_t, a_t)\right] \\
&= \mathbb{E}_{\tau\sim\pi(\tau|s_g)}\left[\sum_{t=0}^{\infty}\gamma^t r_g(s_t, a_t)\right].
\end{aligned}
$$

On the second line, we have changed the bounds of the summation to start at 0, and changed the terms inside the summation accordingly. On the third line, we applied linearity of expectation to move the summation inside the expectation. On the fourth line, we applied linearity of expectation again to move the term for $t = 0$ inside the expectation. Finally, we substituted the definition of $r_g(s, a)$ to obtain the desired result.

B.2 Contrastive RL is Policy Improvement

This section proves the Contrastive RL (NCE) corresponds to policy improvement, yielding policies with higher rewards at each iteration (Lemma 4.2).

Proof. The main idea of the proof is to relate the Q-values for the average policy to the Q-values for the goal-conditioned policy. We do this by employing the result from [31, Appendix C.2], where ϵ is the parameter for filtered relabeling (Sec. 4.5):

$$
\left|Q^{\beta(\cdot mid\cdot, e)}(s, a, e) - Q^{\beta(\cdot|\cdot, e')}(s, a, e)\right| \le \epsilon.
$$

This result means that we are doing policy improvement with approximate Q-values. Then, [10, Lemma 6.1] tells that doing policy improvement using approximate Q-values gives us approximate policy improvement:

$$
\mathbb{E}_{\pi'(\tau|s_g)}\left[\sum_{t=0}^{\infty}\gamma^t r_{s_g}(s_t, a_t)\right] \ge \mathbb{E}_{\pi(\tau|s_g)}\left[\sum_{t=0}^{\infty}\gamma^t r_{s_g}(s_t, a_t)\right] - \frac{2\gamma\epsilon}{1-\gamma} \qquad \text{for all goals } s_g \in \{s_g \mid p_g(s_g) > 0\}.
$$

□

C Contrastive RL (CPC)

In this section, we derive a version of contrastive RL based on the infoNCE objective [95]. Compared with the NCE objective used in contrastive RL (NCE), this objective uses a categorical cross entropy loss instead of a binary cross entropy loss. We replace Eq. 6 with the following infoNCE objective [95]:

$$
\max_f \mathbb{E}_{\substack{(s,a)\sim p(s,a), s_f^{(1)}\sim p^{\pi(\cdot|\cdot)}(s_{t+}|s,a) \\ s_f^{(2:B)}\sim p(s_f)}}\left[\log p^{(1)}\right],
$$

where $p^{(1)}$ is the first coordinate of the softmax over the critic:

$$
p = \textsc{SoftMax}([f(s, a, s_f^{(1)}), \cdots, f(s, a, s_f^{(b)})]).
$$

The optimal critic for the infoNCE loss satisfies [84, 95, 99]

$$
f^*(s, a, s_f) = \log\left(\frac{p^{\pi(\cdot|\cdot)}(s_{t+} = s_f \mid s, a)}{p(s_f)c(s, a)}\right),
$$

where $c(s, a)$ is an arbitrary function. Thus, there are many optimal critics. Choosing actions that maximize the critic f^* does not necessarily correspond to choosing actions that maximize the probability of the future state. Thus, we need to regularize $c(s, a)$ so that it does not depend on a. We do this by introducing a regularizer, based on [123]:

$$\min_f E_{s_f^{(1:B)} \sim p(s_f)} \textsc{LogSumExp}([f(s, a, s_f^{(1)}), \cdots, f(s, a, s_f^{(b)})])^2.$$

To provide some intuition for this regularizer, consider applying this regularizer to an optimal critic:

$$\begin{aligned}
&\textsc{LogSumExp}([f^*(s, a, s_f^{(1)}), \cdots, f^*(s, a, s_f^{(b)})])^2 \\
&= \left(\log \frac{1}{c(s, a)} \sum_{s_f} \frac{p^{\pi(\cdot|\cdot)}(s_{t+} = s_f \mid s, a)}{p(s_f) c(s, a)} \right)^2 \\
&= \left(\log \sum_{s_f \in s_f^{(1:B)}} \frac{p^{\pi(\cdot|\cdot)}(s_{t+} = s_f \mid s, a)}{p(s_f)} - \log c(s, a) \right)^2 \\
&\approx \left(\log \sum_{s_f \in s_f^{(2:B)}} \frac{p^{\pi(\cdot|\cdot)}(s_{t+} = s_f \mid s, a)}{p(s_f)} - \log c(s, a) \right)^2 \\
&\approx \left(\log \mathbb{E}_{s_f \sim p(s_f)} \left[\frac{p^{\pi(\cdot|\cdot)}(s_{t+} = s_f \mid s, a)}{p(s_f)} \right] - \log c(s, a) \right)^2 \\
&= \left(-\log c(s, a) \right)^2.
\end{aligned}$$

In the third line we ignore the positive term; this is reasonable if the batch size is large enough. In the third line we replaced the sum with an expectation; this is biased because $\log(\cdot)$ is not a linear function. Thus, this regularizer (approximately) regularizes $c(s, a)$ to be close to 1 for all states and actions. By reducing the dependency of $c(s, a)$ on the actions a, we can ensure that actions that maximize the critic do maximize the probability of reaching the desired goal. In practice, we add this regularizer with the infoNCE objective, using a coefficient of 1e-2 on the regularizer.

D Contrastive RL (NCE + C-learning)

In this section we describe `contrastive RL (NCE + C-learning)` the combined NCE + C-learning method used in Sec. 5.3 (Fig. 5). Mathematically, the NCE + C-learning objective is a simple, unweighted sum of the C-learning objective and the NCE objective:

$$\begin{aligned}
\mathcal{L}(f) =& (1-\gamma) \mathbb{E}_{(s,a) \sim p(s,a), s_f^+ \sim p(s_{t+1}|s_t,a_t)} [\log \sigma(f(s, a, s_f^+))] \\
&+ \gamma \mathbb{E}_{\substack{s_g \sim p_g(s_g), (s_t,a_t) \sim p(s,a), \\ s_{t+1} \sim p(s_{t+1}|s_t,a_t), a_{t+1} \sim \pi(a_{t+1}|s_{t+1}, s_g)}} \Bigg[\underbrace{\frac{p(s_{t+} = s_g \mid s_t, a_t)}{p(s_f = s_g)}}_{\approx \exp(f(s_{t+1}, a_{t+1}, s_g))} \log \sigma(f(s, a, s_f = s_g)) \Bigg] \\
&+ \mathbb{E}_{s_g \sim p_g(s_g), (s,a) \sim p(s,a)} [\log(1 - \sigma(f(s, a, s_g))] \\
&+ \mathbb{E}_{(s,a) \sim p(s,a), s_f^+ \sim p(s_{t+}|s_t,a_t)} [\log \sigma(f(s, a, s_f^+))] + \mathbb{E}_{(s,a) \sim p(s,a), s_f^- \sim p(s_f)} [\log(1 - \sigma(f(s, a, s_f^-)))].
\end{aligned}$$

While we could use half the batch to compute each of the loss terms, we can increase the effective sample size by being careful with how the terms are estimated. First, we note that the first two terms of each loss are similar – sample a future state (either the next state or a future state) and label it as a positive. We can thus combine these two terms by sampling from a mixture of these two distributions,

$$\tilde{p}(s_f \mid s_t, a_t) = \frac{1-\gamma}{1+1-\gamma} p(s_{t+1} = s_f \mid s_t, a_t) + \frac{1}{1+1-\gamma} p(s_{t+} = s_f \mid s_t, a_t),$$

and scaling the resulting loss by $1 + 1 - \gamma = 2 - \gamma$:

$$\begin{aligned}
\mathcal{L}_1(f) \triangleq& (1-\gamma) \mathbb{E}_{(s,a) \sim p(s,a), s_f^+ \sim p(s_{t+1}|s_t,a_t)} [\log \sigma(f(s, a, s_f^+))] + + \mathbb{E}_{(s,a) \sim p(s,a), s_f^+ \sim p(s_{t+}|s_t,a_t)} [\log \sigma(f(s, a, s_f^+))] \\
&= (2-\gamma) \mathbb{E}_{(s,a) \sim p(s,a), s_f^+ \sim \tilde{p}(s_f|s_t,a_t)} [\log \sigma(f(s, a, s_f^+))]
\end{aligned}$$

This trick increases the effective sample size by 96% (130 $\rightarrow$ 256, as measured using [64]).

Both losses also contain terms that are an expectation over random goals. We can likewise combine those terms:

$$
\begin{aligned}
\mathcal{L}_2(f) \triangleq & \gamma \mathbb{E}_{\substack{s_g \sim p_g(s_g), (s_t, a_t) \sim p(s,a), \\ s_{t+1} \sim p(s_{t+1}|s_t,a_t), a_{t+1} \sim \pi(a_{t+1}|s_{t+1}, s_g)}} \left[\lfloor \exp(f(s_{t+1}, a_{t+1}, s_g)) \rfloor_{\text{sg}} \log \sigma(f(s, a, s_f = s_g)) \right] \\
& + \mathbb{E}_{s_g \sim p_g(s_g), (s,a) \sim p(s,a)} \left[\log(1 - \sigma(f(s, a, s_g)) \right] + \mathbb{E}_{(s,a) \sim p(s,a), s_f^- \sim p(s_f)} [\log(1 - \sigma(f(s, a, s_f^-)))] \\
& = \gamma \mathbb{E}_{\substack{s_g \sim p_g(s_g), (s_t, a_t) \sim p(s,a), \\ s_{t+1} \sim p(s_{t+1}|s_t,a_t), a_{t+1} \sim \pi(a_{t+1}|s_{t+1}, s_g)}} \left[\lfloor \exp(f(s_{t+1}, a_{t+1}, s_g)) \rfloor_{\text{sg}} \log \sigma(f(s, a, s_f = s_g)) \right] \\
& + 2 \mathbb{E}_{s_g \sim p_g(s_g), (s,a) \sim p(s,a)} \left[\log(1 - \sigma(f(s, a, s_g)) \right].
\end{aligned}
$$

Note that estimating the first term in $\mathcal{L}_2$ requires sampling an action for each next state and goal pair. This prohibits us from using the same outer product trick as in Sec. 4.4 to estimate this term. While we could still use that trick to estimate the second term in $\mathcal{L}_2$, we found that doing so hurt performance. We hypothesize that the reason is that this creates an imbalance in the gradients – some goals are labeled as negatives but are not also labeled as positives. Thus, we do not use the outer product trick for this method. The final objective is $\mathcal{L}(f) = \mathcal{L}_1(f) + \mathcal{L}_2(f)$.

E Experimental Details

We implemented contrastive RL and the baselines using the ACME RL library [57] in combination with JAX [13]. Precisely, we took the SAC agent[9] and made the modifications below. Unless otherwise mentioned, we used the same hyperparameters as this implementation.

1. Implemented environments that returned observations that contained the original observation concatenated with the goal. While the goals are resampled when sampling from the replay buffer, assuming that observations include the goals means that the Q-function and policy networks do not need to include an additional input for the goal.

2. Modified the replay buffer to use trajectories rather than transitions. This allows us to sample (s_t, a_t, s_f) triplets.

3. Modified the critic network to be parametrized as an inner product between state-action representations and goal representations.

4. Modified the critic loss based on Alg. 1. Note that the actor loss does not need to be modified (except for removing the entropy term for state-based tasks).

We summarize the hyperparameters in Table 2. Both the state-action encoder and the goal encoder are fully-connected neural networks with 2 layers of size 256 with ReLU activations. We found that normalizing the final representation, applying a final activation function, or using a learnable temperature hurt performance, so we do not use these tricks. For image-based tasks, observations have size $(64, 64, 3)$, and we use a CNN encoder from prior work [88] to encode the observations before passing them to the encoders. The policy has a similar architecture: an image-encoder for image-based tasks, followed by 2 fully connected layers with size 256 and ReLU activations.

Table 2: Hyperparameters for our method and the baselines.

hyperparameter	value
batch size	256
learning rate	3e-4 for all components
discount	0.99
actor target entropy	0 for state-based experiments, $-\dim(a)$ for image-based experiments
target EMA term (for TD3 and SAC)	0.005
image encoder architecture	Taken from Mnih et al. [88]
image decoder architecture (for auto-encoder and model-based baselines)	Taken from Ha and Schmidhuber [47]
hidden layers sizes (for actor and representations)	(256, 256)
initial random data collection	10,000 transitions
replay buffer size	1,000,000 transitions
samples per insert[1]	256
train-collect interval[2]	16 for state-based tasks, 64 for image-based tasks
representation dimension $(\dim(\phi(s,a)), \dim(\psi(s_g)))$	64
actor minimum std dev	1e-6
number of augmentations (for DrQ only)	4
logsumexp regularizer coefficient (for CPC only)	1e-2
action repeat	None
goals for actor loss	random states (not future states)

[1] How many times is each transition used for training before being discarded.
[2] We collect N transitions, add them to the buffer, and then do N gradient steps using the experience sampled randomly from the buffer.

[9] `https://github.com/deepmind/acme/tree/master/acme/agents/jax/sac`

Table 3: Changes to hyperparameters for offline RL experiments (Fig. 1).

hyperparameter	value
batch size	$256 \rightarrow 1024$
representation dimension	$64 \rightarrow 16$
hidden layers sizes (for actor and representations)	$(256, 256) \rightarrow (1024, 1024)$, as in [25].
goals for actor loss	future states

E.1 Environments

fetch reach (image, state) – This task is taken from Plappert et al. [97]. This task involves moving a robotic arm to a goal location in free space. The benchmark specifies success as reaching within 5cm of the goal.

fetch push (image, state) – This task is taken from Plappert et al. [97]. This task involves using a robotic arm to push an object across a table to a goal location. The benchmark specifies success as reaching within 5cm of the goal.

sawyer push (image, state) – This task is taken from Yu et al. [139]. This task involves using a robotic arm to push an object across table to a goal location. The benchmark specifies success as reaching within 5cm of the goal.

ant umaze (state) – This task is taken from Fu et al. [36]. This task involves controlling an ant-like robot towards a goal location, which is sampled randomly in a "U"-shaped maze. Unlike all other tasks in this paper, the goal was lower dimensional than the observation: the goal was just the XY coordinate of the desired position. Following prior work [15, 115], success is defined as reaching within 0.5m of the goal.

sawyer bin (image, state) – This task is taken from Yu et al. [139]. This task involves using a robotic arm to pick up a block from one bin and place it at a goal location in another bin. The benchmark specifies success as reaching within 5cm of the goal.

point Spiral11x11 (image) – This task is an image-based version of the 2D navigation tasks in Eysenbach et al. [30]. This task involves directly controlling the XY coordinates of an agent to reach a goal in a spiral-shaped maze (see Fig. 16). We define success as reaching within 2m of the goal.

F Additional Experiments

F.1 Linear regression with the learned features

To study the learned representations in isolation we take the state-action representations $\phi(s, a)$ trained on the image-based `point NineRooms` task, and run a linear probe [3, 51] experiment to see whether the representations have learned to encode task-relevant information (the shortest path distance to the goal).

We use the task of nine-room navigation and run Contrastive RL and TD3+HER on it. We visualize the environment in Fig. 8a and the agent randomly initialized in one of the nine rooms is commanded to go to the goal position. We dump the replay buffer during training as the dataset and run a linear regression to predict the shortest distance between the agent and the goal. Note that this shortest path distance is not the Euclidean distance since there are walls blocking the way. Fig. 8b shows that features learned by contrastive RL can predict this distance better than all baselines.

As shown in Fig. 8b, contrastive RL (NCE) learns representations that achieve lower test error than those learned by TD3+HER and by a random CNN encoder.

F.2 When is contrastive learning better than learning a foreward model?

In Fig. 2a, we observed that the model-based baseline performed well on the `ant umaze` task, but poorly on many of the other tasks. One explanation is that the model-based approach will perform well when the goal is relatively low-dimensional, and that contrastive learning will be more useful in settings with higher-dimensional goals. We tested this experiment on the 7-dimensional `sawyer push` environment. We applied both contrastive RL and the model-based baseline to versions of this task where the goal was varied from 1-dimensional to 7-dimensional. Note that changing the goal dimension changes the task: a 1-dimensional goal corresponds to moving the gripper to the correct X position, whereas a 7-dimensional goal corresponds to moving the object and gripper to the correct poses. We measured the Euclidean distance to the goal (↓ is better). We show results in Fig. 9. As expected, higher-dimensional goals are a bit more challenging to achieve. What we are really interested in is the *gap* between the model-based approach and the contrastive RL, which opens up starting with a 4-dimensional goal. Altogether, this

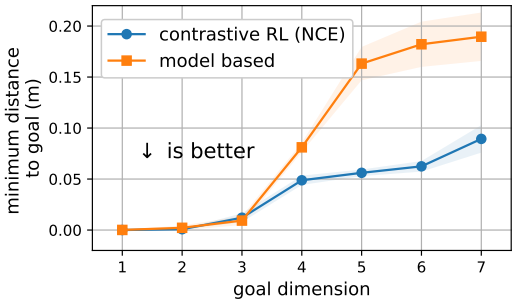
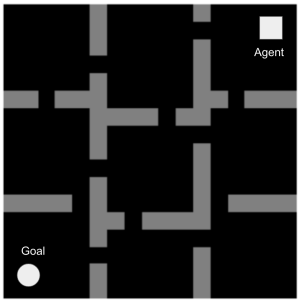
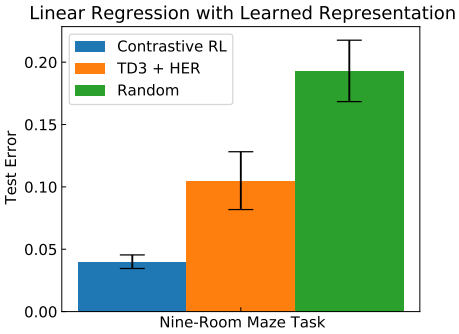


Figure 9: Contrasive learning outperforms a forward model when the goal is 4-dimensional or larger. Error bars show the standard deviation across 5 random seeds.



(a) **Nine-Room environment.**



(b) **Linear probe experiment.**

Figure 8: **Linear regression with the learned features.** Contrastive RL can produce better features for predicting the shortest-path distance, indicating that the learned features have captured highly non-linear information about the environment dynamics.

experiment provides some evidence that contrastive RL may be preferred over a forward model, even for tasks with very low dimensional goals.

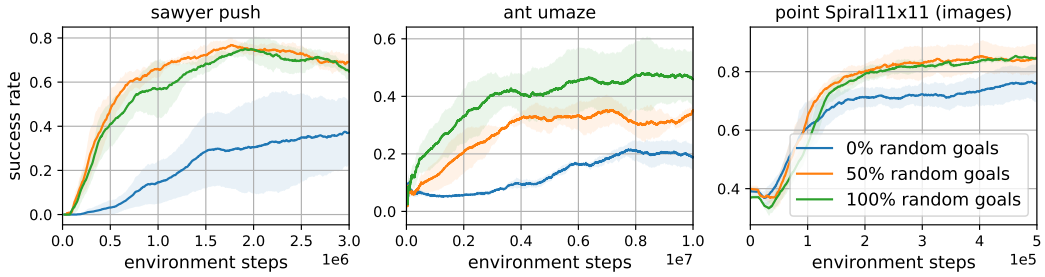


Figure 10: **Goals used for the actor loss.** Goals are either sampled from the distribution over future states or from a distribution of random states. Error bars show the standard deviation across 5 random seeds.

F.3 Goals used in the actor loss

In theory, the distribution of goals for the actor loss (Eq. 7) does not affect the optimal policy, as long as the distribution has full support. In our experiments, we sampled these goals randomly, in the same way that we sampled negative examples for contrastive learning. We ran an ablation experiment to study this decision, and show results in Fig. 10. These results show that sampling future goals consistently performs poorly, perhaps because it results in only training the policy on how to reach "easy" goals. A mixture of future goals and random goals works much better, but the best results seem to come from training on only random goals.

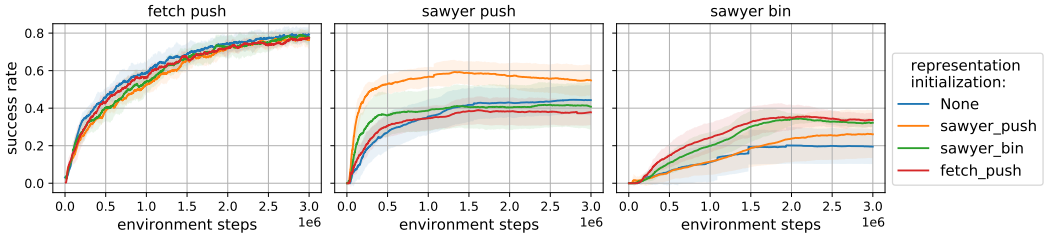


Figure 11: **Transferring representations to solve new tasks.** After training the representations on one task for 1M environment steps, we used them to initialize a new agent for solving a new task.

F.4 Transferring representations to solve new tasks

In this experiment, we studied whether the representations learned by contrastive RL (NCE) for one task might be useful for solving another task. We started by training contrastive RL (NCE) on three image-based tasks: `fetch push`, `sawyer push`, and `sawyer bin`. The observations for all tasks look different and the sawyer and fetch tasks have different robots. The two sawyer tasks look the most similar because they both come from the metaworld [139] benchmark suite. We used the representations learned for each of these tasks to initialize a second contrastive RL agent, which we used to solve this same set of tasks. We were primarily interested in transfer – do the representations from one task help in learning to solve another task? Intuitively, even if the tasks are different, a good representation will capture some structural properties (e.g., identifying the robot arm, and identifying objects), which should transfer across the task.

We show results in Fig. 11. After training on the first task for 1M environment steps, we used the learned representation as initialization for solving the new task. On the `fetch push` task, we see little benefit from using pretrained representations, perhaps because the task is relatively easy. On the `sawyer push`, we see the largest benefit from pretraining the representations on the same task as the target task. More interestingly, we see a small benefit from taking the representations learned on the `sawyer bin` task and using those to solve the `sawyer push`. On the most challenging task, `sawyer bin`, using representations pretrained on either `fetch push` or `sawyer bin` can accelerate

the solving of this task. Taken together, these results suggest that transferring the representations from one task to another is sometimes useful.

F.5 Robustness to Environment Perturbations

We ran an preliminary experiment to study whether the image-based policies learned by contrastive RL (NCE) were robust to perturbations in the environment. We took an agent trained on the `fetch push` with image-observations, and evaluated the agent on four variants of the environment (see Fig. 12):

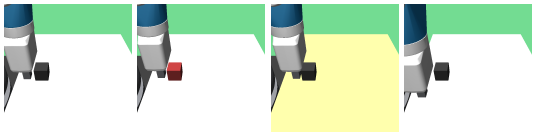


Figure 12: Perturbations to the image-based `fetch push` environment.

- Original environment, without modification;
- Object color changed from black to red;
- Table color changed from white to yellow;
- Initial arm position moved towards the camera.

In each setting, we evaluate the success rate over 20 trials, and repeated 5 times to compute standard deviations (for a total of 100 trials). The learned agent was robust to the object color, with the success rate changing from $78 \pm 5\%$ to $73 \pm 10\%$. The agent was also robust to the change in initial position ($87 \pm 6\%$). However, changing the table color caused the agent to fail ($0 \pm 0\%$), perhaps because the table color consumes a large fraction of the image pixels.

F.6 Additional figures

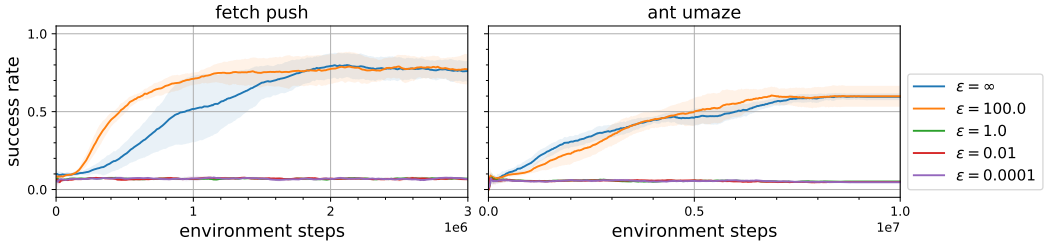


Figure 13: **Filtered relabeling.** We filter the relabeled experience so that the agent only trains on experience where the probability under the commanded goal is similar to the probability under the actually-reached goal. While such filtering is required to prove convergence, these results suggest that good performance can be achieved without this filtering step.

This section presents additional figures.

- Fig. 13 compares contrastive RL (NCE) with varying values of the filtering parameter ϵ, described in Sec. 4.5.
- Fig. 14 – This plot shows a TSNE embedding of the state-action representations $\phi(s, a)$ for one trajectory of the `bin picking` task. This experiment uses image observations.
- Fig. 15 – This plot shows a TSNE embedding of the state-action representations from the same `bin picking` task. We sampled states and actions using a trained agent. After computing the TSNE embedding, we used RasterFairy [65] to rectify the embeddings to a grid.
- Fig. 16 – A TSNE embedding of image representations from the `point Spiral11x11` task.
- Fig. 17 – Using the same representations for the `point Spiral11x11` task, we measure the similarity between the critic gradients when evaluated at the same state but different goals, $\langle \frac{\partial f}{\partial s}|_{(s,g)}, \frac{\partial f}{\partial s}|_{(s,g')} \rangle$.

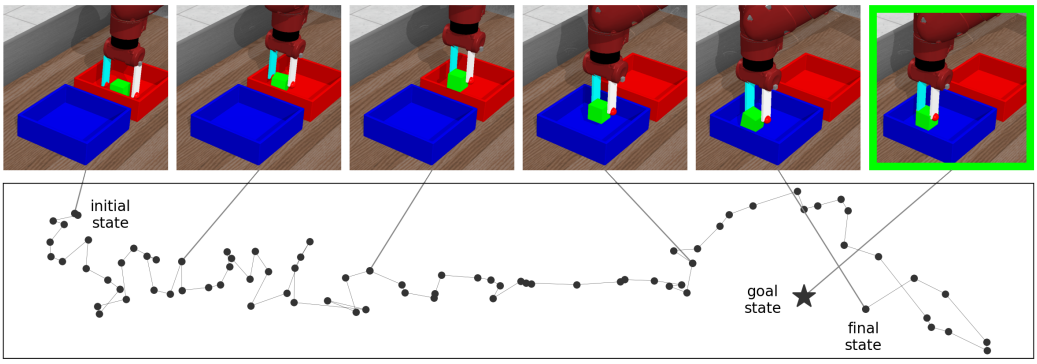


Figure 14: **Visualizing the learned representations.** *(Top)* We show five observations from the bin picking task, as well as the goal image. *(Bottom)* A TSNE embedding of the image representations $\phi(s, a)$ learned by Contrastive RL (NCE). Note that different parts of the task (e.g., reaching, picking, placing) are well separated in the learned representation space.

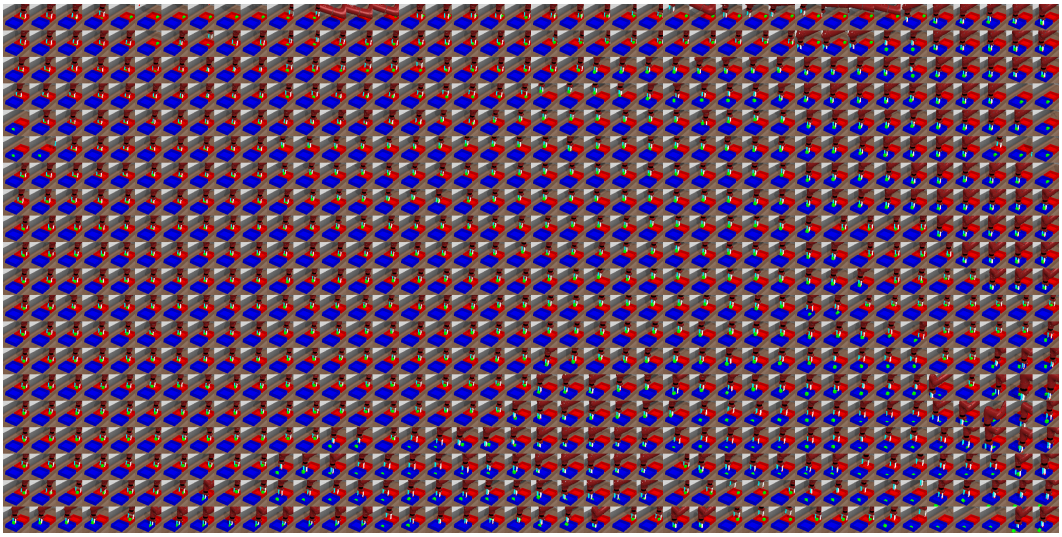
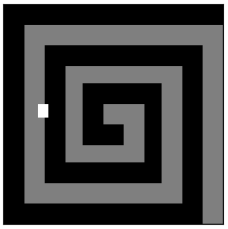
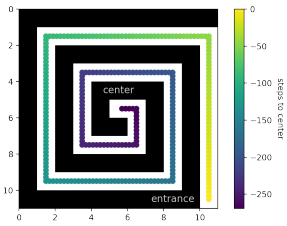


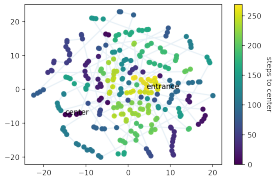
Figure 15: Visualizing the image representations learned by our method on the `sawyer bin`. We compute a TSNE embedding of the representations, and then align the embeddings to a grid using RasterFairy [65].



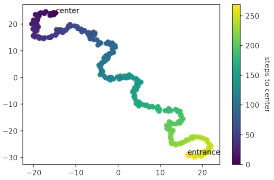
(a)



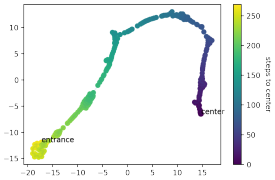
(b)



(c) untrained encoder

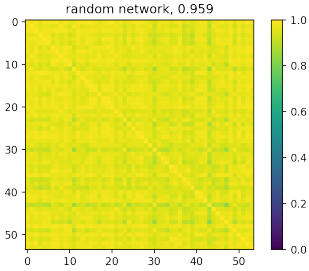


(d) contrastive RL (NCE)

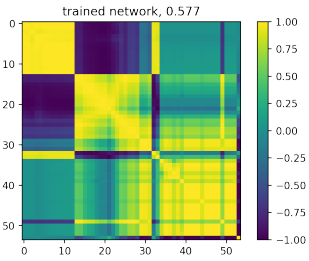


(e) TD3 + HER

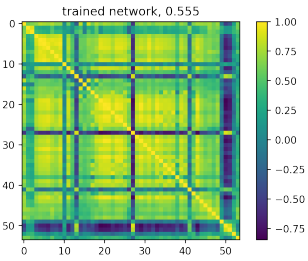
Figure 16: **TSNE embedding of representations** $\phi(s, a)$**.** *(a)* Using the `point Spiral11x11` task, *(b)* we generated image observations at 270 locations throughout the maze. We computed the state-action representations of these images, using action = (0, 0). *(c, d, e)* A TSNE embedding of these representations reveals that the untrained encoder does not capture the structure of the environment, whereas both our method and the TD3 + HER baseline do capture the maze structure.



(a) Random neural network



(b) C-learning



(c) contrastive RL (NCE)

Figure 17: **Analyzing the gradients.** We plot the cosine similarity between the (normalized) gradients of the critic function with respect to the goal images. An untrained network has high gradient similarity, meaning that updates to one state/task affect the networks predictions at many other states/tasks, a phenomenon that prior work has identified as being detrimental to learning [2, 70, 135, 138]. Our method converges to a network where gradients at one state have a low similarity with gradients at other states. A similar plot showing gradients with various state inputs shows a similar effect.

G Failed Experiments

1. *Representation normalization*: We experimented with many ways of normalizing the learned representations, including L2 normalization, scaling the representations by a learned temperature parameter, and applying an elementwise tanh activation. None of these modifications consistently improved performance.
2. *Momentum encoder*: Prior contrastive learning methods have found it useful to use a target encoder or momentum buffer. We experimented with many similar tricks, including using a momentum buffer for goal representations, sampling some fraction of goal representations from a fixed random distribution, increasing the learning rate for the state encoder's final layer, and decreasing the learning rate for all layers in the goal encoder. None of these tricks consistently improved performance.
3. *Frame stacking* – This tended to decrease performance slightly.
4. *Loss scaling* – Our contrastive RL (NCE) method uses the negative label much more often than the positive label. We tried scaling the loss terms so that the negative and positive examples received the same total weight, but found this had no effect on performance.