PROBABILISTIC REINFORCEMENT LEARNING: USING DATA TO DEFINE DESIRED OUTCOMES AND INFERRING HOW TO GET THERE

BENJAMIN EYSENBACH

July 2023
CMU-ML-23-103

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Ruslan Salakhutdinov (Chair)
Sergey Levine
Jeff Schneider
Leslie Kaelbling

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

This research was funded by the National Science Foundation award DGE1745016, a graduate fellowship from Google and a graduate fellowship from the Hertz Foundation.

Keywords: reinforcement learning, machine learning, probability inference, decision making, planning, goal reaching, goal conditioned, goal directed, outcomes, data-driven, probability

PUBLICATIONS

The completed work in this thesis proposal is primarily based on the following work. Additional completed work is cited inline.

[1] Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. "Mismatched No More: Joint Model-Policy Optimization for Model-Based RL." In: *Advances in Neural Information Processing Systems*. 2022.

[2] Benjamin Eysenbach, Sergey Levine, and Ruslan Salakhutdinov. "Replacing Rewards with Examples: Example-Based Policy Search via Recursive Classification." In: *Advances in Neural Information Processing Systems* 35 (2021).

[3] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. "Search on the replay buffer: Bridging planning and reinforcement learning." In: *Advances in Neural Information Processing Systems*. 2019, pp. 15246–15257.

[4] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. "C-Learning: Learning to Achieve Goals via Recursive Classification." In: *International Conference on Learning Representations*. 2021.

[5] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. "Contrastive Learning as Goal-Conditioned Reinforcement Learning." In: *Advances in Neural Information Processing Systems*. 2022.

[6] Raj Ghugare, Homanga Bharadhwaj, Benjamin Eysenbach, Sergey Levine, and Russ Salakhutdinov. "Simplifying Model-based RL: Learning Representations, Latent-space Models, and Policies with One Objective." In: *International Conference on Learning Representations*. 2023.

[7] Tianjun Zhang, Benjamin Eysenbach, Ruslan Salakhutdinov, Sergey Levine, and Joseph E. Gonzalez. "C-Planning: An Automatic Curriculum for Learning Goal-Reaching Tasks." In: *International Conference on Learning Representations*. 2022.

ACKNOWLEDGMENTS

The work contained in this thesis is the result of five years of collaborations, with a fantastic team of students and researchers. I have truly enjoyed these collaborations: they have not only improved the work, but also made me a better researcher. Thanks to my family for their continual support throughout the PhD. Thanks to my advisors for teaching me how to do research, and for letting me explore some strange corners of the ML ecosystem.

CONTENTS

LIST OF TABLES

1 INTRODUCTION

This thesis studies algorithms for teaching autonomous agents to complete tasks through trial and error learning. Typically, this problem is posed as a reinforcement learning (RL) problem, wherein agents attempt to maximize a user-provided reward function. The algorithms studied here take a different approach, largely eschewing the reward function and instead learning to achieve desired outcomes directly from data. This approach allows users to employ algorithmic tools from the supervised and unsupervised learning, while also surfacing an interface that allows non-expert users to teach agents new tasks.

The main challenge in the design of these methods is predicting the probability of desired outcomes, especially when the outcomes only occur hundreds of steps into the future, and especially when using off-policy data. To this end, the first part of this thesis develops an algorithm based on recursive classification that estimates the probability of future states via a temporal difference update (Chapter 2). This method is directly applicable to environments with continuous states and actions, does not require any hand-crafted distance metrics, and leads to an algorithm for goal-conditioned RL that outperforms prior methods. We then generalize this idea to tasks that can be solved in many ways, allowing more flexible task specification and providing broader generalization capabilities.

While framing control problems in terms of desired outcomes provides an easy mechanism to specify *what* the task is, it leaves no room for specifying *how* the task should be solved, raising the question of whether these methods are restricted to simple tasks. To lift this limitation, we consider inferring the *structure* of solutions to complex tasks. Because the algorithms introduced in the first part are probabilistic in nature, it is easy to incorporate this structure as an unobserved latent variable. These new algorithms infer this task structure; in doing so, they decompose the control problem into a series of easier problems, thereby accelerating learning. We first discuss the goal-conditioned setting, where this inferential perspective leads to a simple and theoretically justified method for integrating goal-conditioned RL into classical planning pipelines (Chapter 4). RL is used to estimate distances and learn a local policy, while graph search over observations (e.g., images) determines the high-level path to the goal. This approach substantially outperforms standard goal-conditioned RL algorithms. We then consider a different way of structuring the task solution: as a composition of a learned dynamics model and policy (Chapter 5). The result is an algorithm for model-

based RL where the model and policy are jointly optimized using the same objective, which is a lower bound on expected returns.

This thesis builds upon the work presented in the initial thesis proposal in two primary directions. **First**, we have explored a geometric interpretation of recursive classification (Chapter 2), drawing a close connection between representation learning and reinforcement learning (Chapter 3). This connection has allowed us to extend recursive classification to tasks specified post-hoc via a limited number of reward-labeled states, and has allowed us to apply these methods to real-world, image-based robotic manipulation tasks. **Second**, we have extended the latent-variable perspective of RL (Chapters 4 and 5) to perform inference over learned representations (Sec. 5.6). This extension enables our approach to scale to higher dimensional tasks and provides a substantial computational speedup.

Part I

DATA-DIRECTED DECISION-MAKING

Reinforcement learning is typically studied through an the lens of utility maximization, a lens that makes transparent the field's roots in behavioral science and explains why much RL research is conducted in operations research departments. However, this lens obscures the role of *data* in decision making. RL is increasingly studied by ML researchers, who take data as their bread and butter, resulting in methods and analysis that often inadvertantly highlight the disconnect between data and the standard MDP formalism. It is easy to imagine a student in an introductory RL class asking "What does RL have to do with ML at all?"

In this Part, we describe decision-making algorithms that treat data as a first-class citizen: data (not rewards) will determine what makes an outcome good. We will focus on learning goal-directed behavior, one of the long-standing and important problems in AI and RL. Here, tasks will be specified not with a scalar reward function, but instead by a single observation of a goal state. Sec. 2.7 will describe how similar ideas can solve fully-general RL problems (e.g., by providing multiple observations (data!) of good outcomes).

Our algorithms for learning goal-directed behavior are one form of unsupervised pretraining: like pretraining methods in NLP and computer vision, our methods learn primitives (representations) without the need for (reward) labels. However, there are alternative forms of unsupervised pretraining for RL, often taking the form of offline RL (which requires reward labels), representation learning, and model-learning. In Chapter 3, we extend our algorithms for goal-directed behavior in a way that highlights the close connections between all these forms of unsupervised pretraining. By parametrizing our goal-directed algorithm in a certain way, we will see that value estimation (as done in offline RL), representation learning, and (implicit) model learning are three ways of interpreting a single method. Beyond providing excellent empirical results, this work may provide guidance on how algorithms and ideas might be shared and unified across the RL community.

2

LEARNING TO ACHIEVE GOALS VIA RECURSIVE CLASSIFICATION

2.1 INTRODUCTION

In this work, we aim to reframe the goal-conditioned reinforcement learning (RL) problem as one of *predicting* and *controlling* the future state of the world. This reframing is useful not only because it suggests a new algorithm for goal-conditioned RL, but also because it explains a commonly used heuristic in prior methods, and suggests how to automatically choose an important hyperparameter. The problem of predicting the future amounts to learning a probability density function over future states, agnostic of the time that a future state is reached. The future depends on the actions taken by the policy, so our predictions should depend on the agent's policy. While we could simply witness the future, and fit a density model to the observed states, we will be primarily interested in the following prediction question: Given experience collected from one policy, can we predict what states a *different* policy will visit? Once we can predict the future states of a different policy, we can *control* the future by choosing a policy that effects a desired future.

While conceptually similar to Q-learning, our perspective is different in that we make no reliance on reward functions. Instead, an agent can solve the prediction problem *before* being given a reward function, similar to models in model-based RL. Reward functions can require human supervision to construct and evaluate, so a fully autonomous agent can learn to solve this prediction problem before being provided any human supervision, and reuse its predictions to solve many different downstream tasks. Nonetheless, when a reward function is provided, the agent can estimate its expected reward under the predicted future state distribution. This perspective is different from prior approaches. For example, directly fitting a density model to future states only solves the prediction problem in the on-policy setting, precluding us from predicting where a different policy will go. Model-based approaches, which learn an explicit dynamics model, do allow us to predict the future state distribution of different policies, but require a reward function or distance metric to learn goal-reaching policies for controlling the future. Methods based on temporal difference (TD) learning [229] have been used to predict the future state distribution [18, 41, 233] and to learn goal-reaching policies [110, 202]. Section 2.3 will explain why these approaches do not learn a true Q function in continuous environments with sparse rewards, and

it remains unclear what the learned Q function corresponds to. In contrast, our method will estimate a well defined classifier.

Since it is unclear how to use Q-learning to estimate such a density, we instead adopt a contrastive approach, learning a classifier to distinguish "future states" from random states, akin to Gutmann and Hyvärinen [88]. After learning this binary classifier, we apply Bayes' rule to obtain a probability density function for the future state distribution, thus solving our prediction problem. While this initial approach requires on-policy data, we then develop a bootstrapping variant for estimating the future state distribution for different policies. This bootstrapping procedure is the core of our goal-conditioned RL algorithm.

The main contribution of our work is a reframing of goal-conditioned RL as estimating the probability density over future states. We derive a method for solving this problem, C-learning, which we use to construct a complete algorithm for goal-conditioned RL. Our reframing lends insight into goal-conditioned Q-learning, leading to a hypothesis for the optimal ratio for sampling goals, which we demonstrate empirically. Experiments demonstrate that C-learning more accurately estimates the density over future states, while remaining competitive with recent goal-conditioned RL methods across a suite of simulated robotic tasks.[1]

2.2 RELATED WORK

Common goal-conditioned RL algorithms are based on behavior cloning [43, 53, 76, 85, 152, 174, 228], model-based approaches [49, 167], Q-learning [110, 186, 202], and semi-parametric planning [29, 56, 169, 200]. Most prior work on goal-conditioned RL relies on manually-specified reward functions or distance metric, limiting the applicability to high-dimensional tasks. Our method will be most similar to the Q-learning methods, which are applicable to off-policy data. These Q-learning methods often employ hindsight relabeling [10, 110], whereby experience is modified by changing the commanded goal. New goals are often taken to be a future state or a random state, with the precise ratio being a sensitive hyperparameter. We emphasize that our discussion of goal sampling concerns relabeling previously-collected experience, not on the orthogonal problem of sampling goals for exploration [60, 183, 186].

Our work is closely related to prior methods that use TD-learning to predict the future state distribution, such as successor features [17, 18, 41, 233] and generalized value functions [202, 205, 232]. Our approach bears a resemblance to these prior TD-learning methods, offering insight into why they work and how hyperparameters such as the

1 Project website with videos and code: https://ben-eysenbach.github.io/c_learning/

goal-sampling ratio should be selected. Our approach differs in that it does not require a reward function or manually designed relabeling strategies, with the corresponding components being derived from first principles. While prior work on off-policy evaluation [149, 160] also aims to predict the future state distribution, our work differs is that we describe how to *control* the future state distribution, leading to goal-conditioned RL algorithm.

Our approach is similar to prior work on noise contrastive estimation [88], mutual-information based representation learning [162, 177], and variational inference methods [22, 48, 101, 219, 243]. Like prior work on the probabilistic perspective on RL [112, 137, 178, 194, 238, 240, 275], we treat control as a density estimation problem, but our main contribution is orthogonal: we propose a method for estimating the future state distribution, which can be used as a subroutine in both standard RL and these probabilistic RL methods.

2.3 PRELIMINARIES

We start by introducing notation and prior approaches to goal-conditioned RL. We define a controlled Markov process by an initial state distribution $p_1(\mathbf{s_1})$ and dynamics function $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$. We control this process by a Markovian policy $\pi_\theta(\mathbf{a_t} \mid \mathbf{s_t})$ with parameters θ. We use $\pi_\theta(\mathbf{a_t} \mid \mathbf{s_t}, \mathbf{g})$ to denote a goal-oriented policy, which is additionally conditioned on a goal $\mathbf{g} \in \mathcal{S}$. We use $\mathbf{s_{t+}}$ to denote the random variable representing a future observation, defined by the following distribution:

Definition 1. *The future $\gamma-$discounted state density function is*

$$p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) \triangleq (1-\gamma) \sum_{\Delta=1}^{\infty} \gamma^\Delta p_\Delta^\pi(\mathbf{s_{t+\Delta}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}),$$

where $s_{t+\Delta}$ denotes the state exactly Δ in the future, and constant $(1-\gamma)$ ensures that this density function integrates to 1.

This density reflects the states that an agent would visit if we collected many infinite-length trajectories and weighted states in the near-term future more highly. Equivalently, $p(\mathbf{s_{t+}})$ can be seen as the distribution over terminal states we would obtain if we (hypothetically) terminated episodes at a random time step, sampled from a geometric distribution. We need not introduce a reward function to define the problems of predicting and controlling the future.

In discrete state spaces, we can convert the problem of estimating the future state distribution into a RL problem by defining a reward function $r_{\mathbf{s_{t+}}}(\mathbf{s_t}, \mathbf{a_t}) = \mathbb{1}(\mathbf{s_t} = \mathbf{s_{t+}})$, and terminating the episode when the agent arrives at the goal. The Q-function, which typically

represents the expected discounted sum of future rewards, can then be interpreted as a (scaled) probability *mass* function:

$$Q^\pi(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}) = \mathbb{E}_\pi\left[\sum_t \gamma^t r_{\mathbf{s_{t+}}}(\mathbf{s_t}, \mathbf{a_t})\right] = \sum_t \gamma^t \mathbb{P}_\pi(\mathbf{s_t} = \mathbf{s_{t+}}) = \frac{1}{1-\gamma} p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}).$$

However, in continuous state spaces with some stochasticity in the policy or dynamics, the probability that any state *exactly* matches the goal state is zero.

Remark 1. *In a stochastic, continuous environment, for any policy π the Q-function for the reward function $r_{\mathbf{s_{t+}}} = \mathbb{1}(\mathbf{s_t} = \mathbf{s_{t+}})$ is always zero: $Q^\pi(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}) = 0$.*

This Q-function is not useful for predicting or controlling the future state distribution. Fundamentally, this problem arises because the relationship between the reward function, the Q function, and the future state distribution in prior work remains unclear. Prior work avoids this issue by manually defining reward functions [10] or distance metrics [186, 202, 205, 273]. An alternative is to use hindsight relabeling, changing the commanded goal to be the goal actually reached. This form of hindsight relabeling does not require a reward function, and indeed learns Q-functions that are not zero [145]. However, taken literally, Q-functions learned in this way must be incorrect: they do not reflect the expected discounted reward. An alternative hypothesis is that these Q-functions reflect probability *density* functions over future states. However, this also cannot be true:

Remark 2. *For any MDP with the sparse reward function $\mathbb{1}(\mathbf{s_t} = \mathbf{s_{t+}})$ where the episode terminates upon reaching the goal, Q-learning with hindsight relabeling acquires a Q-function in the range $Q^\pi(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}) \in [0, 1]$, but the probability density function $p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ has a range $[0, \infty)$.*

For example, if the state space is $\mathcal{S} = [0, \frac{1}{2}]$, then there must exist some state s_{t+} such that $Q^\pi(s_t, a_t, s_{t+1}) \leq 1 < p^\pi_+(\mathbf{s_{t+}} = s_{t+} \mid s_t, a_t)$. Thus, Q-learning with hindsight relabeling also fails to learn the future state distribution. In fact, it is unclear what quantity Q-learning with hindsight relabeling optimizes. In the rest of this work, we will define goal reaching in continuous state spaces in a way that is consistent and admits well-defined solutions (Sec. 2.4), and then present a practical algorithm for finding these solutions (Sec. 2.5).

2.4 FRAMING GOAL CONDITIONED RL AS DENSITY ESTIMATION

This section presents a novel framing of the goal-conditioned RL problem, which resolves the ambiguity discussed in the previous

Algorithm 1 Monte Carlo C-learning

Input trajectories $\{\tau_i\}$
Define $p(s,a) \leftarrow \text{Unif}(\{s,a\}_{(s,a)\sim\tau}), p(s_{t+}) \leftarrow \text{Unif}(\{s_t\}_{s_t\sim\tau, t>1})$
while not converged **do**
  Sample $s_t, a_t \sim p(s,a), s_{t+}^{(0)} \sim p(s_{t+}), \Delta \sim \text{Geom}(1-\gamma)$.
  Set goal $s_{t+}^{(1)} \leftarrow s_{t+\Delta}$
  $\mathcal{F}(\theta) \leftarrow \log C_\theta^\pi(F=1 \mid s_t, a_t, s_{t+}^{(1)}) + \log C_\theta^\pi(F=0 \mid s_t, a_t, s_{t+}^{(0)})$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{F}(\theta)$
Return classifier C_θ

section. Our main idea is to view goal-conditioned RL as a problem of estimating the density $p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ over future states that a policy π will visit, a problem that Q-learning does not solve (see Section 2.3). Section 2.5 will then explain how to use this estimated distribution as the core of a complete goal-conditioned RL algorithm.

Definition 2. *Given policy π, the **future state density estimation** problem is to estimate the $\gamma-$discounted state distribution of π: $f_\theta^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) \approx p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$.*

The next section will show how to estimate f_θ^π. Once we have found f_θ^π, we will use it to train a goal-conditioned policy, which will maximize the probability of reaching a commanded goal under this (estimated) discounted state occupancy measure:

$$\max_{\pi(\cdot|\cdot,g)} \mathbb{E}_{\pi(\mathbf{a_t}|\mathbf{s_t},g)}\left[p_+^\pi(\mathbf{s_{t+}} = g \mid \mathbf{s_t}, \mathbf{a_t})\right].$$

In Appendix a.1, we discuss the connections between this "maximum probability" objective and the common stochastic shortest path objective.

2.5 C-LEARNING

We now derive an algorithm (C-learning) for solving the future state density estimation problem (Def. 2). First (Sec. 2.5.1), we assume that the policy is fixed, and present on-policy and off-policy solutions. Based on these ideas, Section 2.5.2 builds a complete goal-conditioned RL algorithm for learning an optimal goal-reaching policy. Our algorithm bears a resemblance to Q-learning, and our derivation makes two hypotheses about when and where Q-learning will work best (Sec. 2.5.3).

2.5.1 *Learning the classifier*

Rather than estimating the future state density directly, we will estimate it indirectly by learning a classifier. Not only is classification

generally an easier problem than density estimation, but also it will allow us to develop an off-policy algorithm in the next section. We will call our approach *C-learning*. We start by deriving an on-policy Monte Carlo algorithm (*Monte Carlo C-learning*), and then modify it to obtain an off-policy, bootstrapping algorithm (*off-policy C-learning*). After learning this classifier, we can apply Bayes' rule to convert its binary predictions into future state density estimates. Given a distribution over state action pairs, $p(\mathbf{s_t}, \mathbf{a_t})$, we define the marginal future state distribution $p(\mathbf{s_{t+}}) = \int p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) p(\mathbf{s_t}, \mathbf{a_t}) d\mathbf{s_t} d\mathbf{a_t}$. The classifier takes as input a state-action pair $(\mathbf{s_t}, \mathbf{a_t})$ together with another state $\mathbf{s_{t+}}$, and predicts whether $\mathbf{s_{t+}}$ was sampled from the future state density $p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ ($F = 1$) or the marginal state density $p(\mathbf{s_{t+}})$ ($F = 0$). The Bayes optimal classifier is

$$p(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}) = \frac{p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})}{p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) + p(\mathbf{s_{t+}})}. \tag{2.1}$$

Thus, using $C^\pi_\theta(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})$ to denote our learned classifier, we can obtain an estimate $f^\pi_\theta(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ for the future state density function using our classifier's predictions as follows:

$$f^\pi_\theta(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) = \frac{C^\pi_\theta(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})}{C^\pi_\theta(F = 0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})} p(\mathbf{s_{t+}}). \tag{2.2}$$

While our estimated density f_θ depends on the marginal density $p(\mathbf{s_{t+}})$, our goal-conditioned RL algorithm (Sec. 2.5.2) will not require estimating this marginal density. In particular, we will learn a policy that chooses the action $\mathbf{a_t}$ that maximizes this density, but the solution to this maximization problem does not depend on the marginal $p(\mathbf{s_{t+}})$.

We now present an on-policy approach for learning the classifier, which we call *Monte Carlo C-Learning*. After sampling a state-action pair $(\mathbf{s_t}, \mathbf{a_t}) \sim p(\mathbf{s_t}, \mathbf{a_t})$, we can either sample a future state $\mathbf{s^{(1)}_{t+}} \sim p^\pi_+(\mathbf{s_{t+}} \mid s_t, \mathbf{a_t})$ with a label $F = 1$, or sample $\mathbf{s^{(0)}_{t+}} \sim p(\mathbf{s_{t+}})$ with a label $F = 0$. We then train the classifier maximize log likelihood (i.e., the negative cross entropy loss):

$$\begin{aligned}\mathcal{F}(\theta) \triangleq & \mathbb{E}_{\substack{\mathbf{s_t}, \mathbf{a_t} \sim p(\mathbf{s_t}, \mathbf{a_t}) \\ \mathbf{s^{(1)}_{t+}} \sim p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})}} [\log C^\pi_\theta(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s^{(1)}_{t+}})] \\ & + \mathbb{E}_{\substack{\mathbf{s_t}, \mathbf{a_t} \sim p(\mathbf{s_t}, \mathbf{a_t}) \\ \mathbf{s^{(0)}_{t+}} \sim p(\mathbf{s_{t+}})}} [\log C^\pi_\theta(F = 0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s^{(0)}_{t+}})].\end{aligned} \tag{2.3}$$

To sample future states, we note that the density $p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ is a weighted mixture of distributions $p(\mathbf{s_{t+\Delta}} \mid \mathbf{s_t}, \mathbf{a_t})$ indicating the future state exactly Δ steps in the future:

$$p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) = \sum_{\Delta=0}^{\infty} p(s_{t+\Delta} \mid \mathbf{s_t}, \mathbf{a_t}) p(\Delta)$$
$$\text{where} \quad p(\Delta) = (1-\gamma)\gamma^\Delta = \textsc{Geom}(\Delta; 1-\gamma),$$

where GEOM is the geometric distribution. Thus, we sample a future state $\mathbf{s_{t+}}$ via ancestral sampling: first sample $\Delta \sim \textsc{Geom}(1-\gamma)$ and then, looking at the trajectory containing $(\mathbf{s_t}, \mathbf{a_t})$, return the state that is Δ steps ahead of $(\mathbf{s_t}, \mathbf{a_t})$. We summarize Monte Carlo C-learning in Alg. 1.

While conceptually simple, this algorithm requires on-policy data, as the distribution $p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ depends on the current policy π *and the commanded goal*. Even if we fixed the policy parameters, we cannot use experience collected when commanding one goal to learn a classifier for another goal. This limitation precludes an important benefit of goal-conditioned learning: the ability to readily share experience across tasks. To lift this limitation, the next section will develop a bootstrapped version of this algorithm that works with off-policy data.

We now extend the Monte Carlo algorithm introduced above to work in the off-policy setting, so that we can estimate the future state density for different policies. In the off-policy setting, we are given a dataset of transitions $(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})$ and a *new* policy π, which we will use to generate actions for the next time step, $\mathbf{a_{t+1}} \sim \pi(\mathbf{a_{t+1}} \mid \mathbf{s_{t+1}})$. The main challenge is sampling from $p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$, which depends on the new policy π. We address this challenge in two steps. First, we note a recursive relationship between the future state density at the current time step and the next time step:

$$\underbrace{p_+^\pi(\mathbf{s_{t+}} = s_{t+} \mid \mathbf{s_t}, \mathbf{a_t})}_{\text{future state density at current time step}} = (1-\gamma) \underbrace{p(\mathbf{s_{t+1}} = s_{t+} \mid \mathbf{s_t}, \mathbf{a_t})}_{\text{environment dynamics}}$$
$$+ \gamma \mathbb{E}_{\substack{p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}), \\ \pi(\mathbf{a_{t+1}} \mid \mathbf{s_{t+1}})}} \Bigg[\underbrace{p_+^\pi(\mathbf{s_{t+}} = s_{t+} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}})}_{\text{future state density at next time step}} \Bigg]. \tag{2.4}$$

We can now rewrite our classification objective in Eq. 2.3 as

$$\mathcal{F}(\theta, \pi) = \mathbb{E}_{\substack{p(\mathbf{s_t}, \mathbf{a_t}),\ p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}), \\ \pi(\mathbf{a_{t+1}} \mid \mathbf{s_{t+1}}),\ p_+^\pi(\mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}})}} \Big[(1-\gamma) \log C_\theta^\pi(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})$$
$$+ \gamma \log C_\theta^\pi(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}) \Big]$$
$$+ \mathbb{E}_{p(\mathbf{s_t}, \mathbf{a_t}),\ p(\mathbf{s_{t+}})} \left[\log C_\theta^\pi(F = 0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}) \right]. \tag{2.5}$$

Algorithm 2 Off-Policy C-learning

Input transitions $\{s_t, a, s_{t+1}\}$, policy π_ϕ
while not converged **do**
  Sample $(s_t, a_t, s_{t+1}) \sim p(s_t, a_t, s_{t+1}), s_{t+} \sim p(s_{t+}),$
    $a_{t+1} \sim \pi_\phi(a_{t+1} \mid s_t, a_t)$
  $w \leftarrow \texttt{stop\_grad}\left(\frac{C^\pi_\theta(F=1|s_{t+1},a_{t+1},s_{t+})}{C^\pi_\theta(F=0|s_{t+1},a_{t+1},s_{t+})}\right)$
  $\mathcal{F}(\theta,\pi) \leftarrow (1-\gamma)\log C^\pi_\theta(F=1|s_t,a_t,s_{t+1})$
    $+\log C^\pi_\theta(F=0|s_t,a_t,s_{t+})$
    $+\gamma w \log C^\pi_\theta(F=1|s_t,a_t,s_{t+})$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{F}(\theta,\pi)$
Return classifier C^π_θ

This equation is different from the Monte Carlo objective (Eq. 2.3) because it depends on the new policy, but it still requires sampling from $p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}})$, which also depends on the new policy. Our second step is to observe that we can estimate expectations that use $p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ by sampling from the marginal $\mathbf{s_{t+}} \sim p(\mathbf{s_{t+}})$ and then weighting those samples by an importance weight, which we can estimate using our learned classifier:

$$w(\mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}}) \triangleq \frac{p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}})}{p(\mathbf{s_{t+}})} = \frac{C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}. \tag{2.6}$$

The second equality is obtained by taking Eq. 2.2 and dividing both sides by $p(\mathbf{s_{t+}})$. In effect, these weights account for the effect of the new policy on the future state density. We can now rewrite our objective by substituting the identity in Eq. 2.6 for the $p(\mathbf{s_{t+}})$ term in the expectation in Eq. 2.5. The rewritten objective is

$$\begin{aligned}\mathcal{F}(\theta,\pi) = \mathbb{E}_{\substack{p(\mathbf{s_t},\mathbf{a_t}),\ p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}),\\ p(\mathbf{s_{t+}}),\ \pi(\mathbf{a_{t+1}}|\mathbf{s_{t+1}})}}[&(1-\gamma)\log C^\pi_\theta(F=1\,|\,\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+1}}) \\ &+\gamma\,\lfloor w(\mathbf{s_{t+1}},\mathbf{a_{t+1}},\mathbf{s_{t+}})\rfloor_{\text{sg}}\log C^\pi_\theta(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}) \\ &+\log C^\pi_\theta(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})].\end{aligned} \tag{2.7}$$

We use $\lfloor\cdot\rfloor_{\text{sg}}$ as a reminder that the gradient of an importance-weighted objective should not depend on the gradients of the importance weights. Intuitively, this loss says that next states should be labeled as positive examples, states sampled from the marginal should be labeled as negative examples, but *reweighted* states sampled from the marginal are positive examples.

ALGORITHM SUMMARY. Alg 2 reviews off policy C-learning, which takes as input a policy and a dataset of *transitions*. At each iteration, we sample a $(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})$ transition from the dataset, a potential future state $\mathbf{s_{t+}} \sim p(\mathbf{s_{t+}})$ and the next action $\mathbf{a_{t+1}} \sim \pi(\mathbf{a_{t+1}} \mid \mathbf{s_{t+1}}, \mathbf{s_{t+}})$. We compute the importance weight using the current estimate from

Algorithm 3 Goal-Conditioned C-learning

Input transitions $\{s_t, a, s_{t+1}\}$
while not converged **do**
  Sample $(s_t, a_t, s_{t+1}) \sim p(s_t, a_t, s_{t+1})$,
  $s_{t+} \sim p(s_{t+}), a_{t+1} \sim \pi(a_{t+1} \mid s_t, a_t, s_{t+})$
  $w \leftarrow \texttt{stop\_grad}\left(\frac{C^\pi_\theta(F=1|s_{t+1},a_{t+1},s_{t+})}{C^\pi_\theta(F=0|s_{t+1},a_{t+1},s_{t+})}\right)$
  $\mathcal{F}(\theta,\pi) \leftarrow (1-\gamma)\log C^\pi_\theta(F=1|s_t,a_t,s_{t+1})$
    $+\log C^\pi_\theta(F=0|s_t,a_t,s_{t+})$
    $+\gamma w \log C^\pi_\theta(F=1|s_t,a_t,s_{t+})$
  $\theta \leftarrow \theta - \eta\nabla_\theta \mathcal{F}(\theta,\pi)$
  $\mathcal{G}(\phi) \leftarrow \mathbb{E}_{\pi_\phi(a_t|s_t,g=s_{t+})}[\log C^\pi_\theta(F=1|s_t,a_t,s_{t+})]$
  $\phi \leftarrow \phi + \eta\nabla_\phi \mathcal{G}(\phi)$
Return policy π_ϕ

the classifier, and then plug the importance weight into the loss from Eq. 2.3. We then update the classifier using the gradient of this objective.

C-LEARNING BELLMAN EQUATIONS. In Appendix a.2.1, we provide a convergence proof for off-policy C-learning in the tabular setting. Our proof hinges on the fact that the TD C-learning update rule has the same effect as applying the following (unknown) Bellman operator:

$$\frac{C^\pi_\theta(F=1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})} = (1-\gamma)\frac{p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})}{p(\mathbf{s_{t+}})} + \gamma \mathbb{E}_{\substack{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}),\\ \pi(\mathbf{a_{t+1}}|\mathbf{s_t})}}\left[\frac{C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}\right]$$

This equation tells us that C-learning is equivalent to maximizing the reward function $r_{\mathbf{s_{t+}}}(\mathbf{s_t}, \mathbf{a_t}) = p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})/p(\mathbf{s_{t+}})$, but does so without having to estimate either the dynamics $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$ or the marginal distribution $p(\mathbf{s_t})$.

2.5.2 *Goal-Conditioned RL via C-Learning*

We now build a complete algorithm for goal-conditioned RL based on C-learning. When learning a goal-conditioned policy, commanding different goals will cause the policy to visit different future states. In this section, we describe how to learn a classifier that predicts the future states of a goal-conditioned policy, and how to optimize the corresponding policy to get better at reaching the commanded goal.

To acquire a classifier for a goal-conditioned policy, we need to apply our objective function (Eq. 2.7) to all policies $\{\pi_\phi(a \mid s, g) \mid g \in \mathcal{S}\}$. We therefore condition the classifier and the policy on the *commanded*

goal $g \in \mathcal{S}$. For learning a goal-reaching policy, we will only need to query the classifier on inputs where $\mathbf{s_{t+}} = g$. Thus, we only need to learn a classifier conditioned on inputs where $\mathbf{s_{t+}} = g$, resulting in the following objective:

$$\mathbb{E}_{\substack{p(\mathbf{s_t}, \mathbf{a_t}),\ p(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t}),\\ p(\mathbf{s_{t+}}),\ \pi(\mathbf{a_{t+1}}|\mathbf{s_{t+1}}, \mathbf{g}=\mathbf{s_{t+}})}} [\underbrace{(1-\gamma) \log C^\pi_\theta(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})}_{(a)} + \underbrace{\log C^\pi_\theta(F = 0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})}_{(b)} + \underbrace{\gamma \lfloor w(\mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}}) \rfloor_{\text{sg}} \log C^\pi_\theta(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})}_{(c)}]. \tag{2.8}$$

The **difference** between this objective and the one derived in Section 2.5.1 (Eq. 2.7) is that the next action is sampled from a goal-conditioned policy. The density function obtained from this classifier (Eq. 2.2) represents the future state density of $\mathbf{s_{t+}}$, given that the policy was commanded to reach goal $g = s_{t+}$: $f^\pi_\theta(\mathbf{s_{t+}} = s_{t+} \mid \mathbf{s_t}, \mathbf{a_t}) = p^\pi_+(\mathbf{s_{t+}} = s_{t+} \mid \mathbf{s_t}, \mathbf{a_t}, g = s_{t+})$.

Now that we can estimate the future state density of a goal-conditioned policy, our second step is to optimize the policy w.r.t. this learned density function. We do this by maximizing the policy's probability of reaching the commanded goal: $\mathbb{E}_{\pi_\phi(\mathbf{a_t}|\mathbf{s_t}, g)} [\log p^\pi(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}} = g)]$. Since $p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}, g = \mathbf{s_{t+}})$ is a monotone increasing function of the classifier predictions (see Eq. 2.2), we can write the policy objective in terms of the classifier predictions:

$$\mathcal{G}(\phi) = \max_\phi \mathbb{E}_{\pi_\phi(\mathbf{a_t}|\mathbf{s_t}, g)} [\log C^\pi_\theta(F = 1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}} = g)]. \tag{2.9}$$

If we collect new experience during training, then the marginal distribution $p(\mathbf{s_{t+}})$ will change throughout training. While this makes the learning problem for the classifier non-stationary, the learning problem for the policy (whose solution is independent of $p(\mathbf{s_{t+}})$) remains stationary. In the tabular setting, goal-conditioned C-learning converges to the optimal policy (proof in Appendix a.2.3).

Algorithm Summary: We summarize our approach, which we call *goal-conditioned C-learning*, in Alg. 3. Instead of learning a Q function, this method learns a future state classifier. The classifier is trained using three types of examples: (a) the classifier is trained to predict $y = 1$ when the goal is the next state; (b) $y = 0$ when the goal is a random state; and (c) $y = w$ when the goal is a random state, where w depends on the classifier's prediction at the next state (see Eq. 2.6). Note that (b) and (c) assign different labels to the same goal and can be combined (see Eq. 2.10 in Sec. 2.5.3). This algorithm is simple to implement by taking a standard actor-critic RL algorithm

and changing the loss function for the critic (a few lines of code). Code to reproduce our experiments is online.[2]

2.5.3 *Implications for Q-learning and Hindsight Relabeling*

Off-policy C-learning (Alg. 2) bears a resemblance to Q-learning with hindsight relabeling, so we now compare these two algorithms to make hypotheses about Q-learning, which we will test in Section 2.6. We start by writing the objective for both methods using the cross-entropy loss, $\mathcal{CE}(\cdot,\cdot)$:

$$F_{\text{C-learning}}(\theta,\pi) = (1-\gamma)\mathcal{CE}(C^\pi_\theta(F \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}}), y=1) \tag{2.10}$$
$$+ (1+\gamma w)\mathcal{CE}\left(C^\pi_\theta(F \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}}), y = \frac{\gamma w}{\gamma w + 1} = \frac{\gamma C^{\pi'}_\theta}{\gamma C^{\pi'}_\theta + (1 - C^{\pi'}_\theta)}\right),$$
$$F_{\text{Q-learning}}(\theta,\pi) = (1-\lambda)\mathcal{CE}(Q^\pi_\theta(\mathbf{s_t}, \mathbf{a_t}, g = \mathbf{s_{t+1}}), y=1) \tag{2.11}$$
$$+ \lambda\mathcal{CE}\left(Q^\pi_\theta(\mathbf{s_t}, \mathbf{a_t}, g = \mathbf{s_{t+}}), y = \gamma Q^\pi_\theta(\mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})\right),$$

where $C'_\theta = C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})$ is the classifier prediction at the next state and where $\lambda \in [0,1]$ denotes the *relabeling ratio* used in Q-learning, corresponding to the fraction of goals sampled from $p(\mathbf{s_{t+}})$. There are two differences between these equations, which lead us to make two hypotheses about the performance of Q-learning, which we will test in Section 2.6. The first difference is how the *predicted* targets are scaled for random goals, with Q-learning scaling the prediction by γ while C-learning scales the prediction by $\gamma/(\gamma C'_\theta + (1 - C'_\theta))$. Since Q-learning uses a smaller scale, we make the following hypothesis:

Hypothesis 1. *Q-learning will predict **smaller** future state densities and therefore **underestimate** the true future state density function.*

This hypothesis is interesting because it predicts that prior methods based on Q-learning will not learn a proper density function, and therefore fail to solve the future state density estimation problem. The second difference between C-learning and Q-learning is that Q-learning contains a tunable parameter λ, which controls the ratio with which next-states and random states are used as goals. This ratio is equivalent to a weight on the two loss terms, and our experiments will show that Q-learning with hindsight relabeling is sensitive to this parameter. In contrast, C-learning does not require specification of this hyperparameter. Matching the coefficients in the Q-learning loss (Eq. 2.11) with those in our loss (Eq. 2.10) (i.e., $[1-\lambda, \lambda] \propto [1-\gamma, 1+\gamma w]$), we make the following hypothesis:

Hypothesis 2. *Q-learning with hindsight relabeling will most accurately solve the future state density estimation problem (Def. 2) when random future states are sampled with probability $\lambda = \frac{1+\gamma}{2}$.*

2 https://github.com/google-research/google-research/tree/master/c_learning

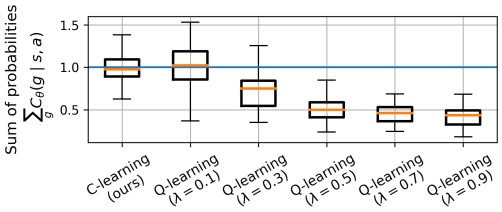


Figure 2.1: **Testing Hypothesis 1**: As predicted, Q-values often sum to less than 1.

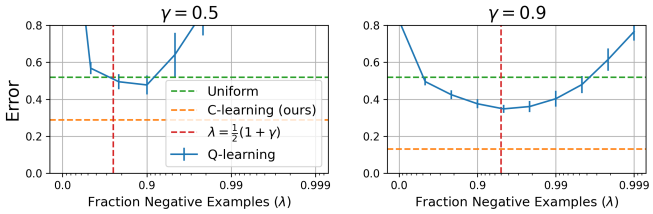


Figure 2.2: **Testing Hypothesis 2**: The performance of Q-learning is sensitive to the relabeling ratio. Our analysis predicts that the optimal relabeling ratio is approximately $\lambda = \frac{1}{2}(1+\gamma)$. C-learning (dashed orange) does not require tuning this ratio and outperforms Q-learning, even when the relabeling ratio for Q-learning is optimally chosen.

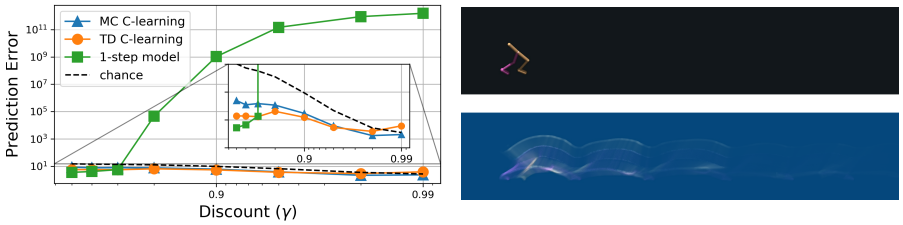
Prior work has found that this goal sampling ratio is a sensitive hyperparameter [10, 186, 273]; this hypothesis is useful because it offers an automatic way to choose the hyperparameter. The next section will experimentally test these hypotheses.

2.6 EXPERIMENTS

We aim our experiments at answering the following questions:

1. Do Q-learning and C-learning accurately estimate the future state density (Problem 2)?
2. (Hypothesis 1) Does Q-learning underestimate the future state density function (§ 2.5.3)?
3. (Hypothesis 2) Is the predicted relabeling ratio $\lambda = (1+\gamma)/2$ optimal for Q-learning (§ 2.5.3)?
4. How does C-learning compare with prior goal-conditioned RL methods on benchmark tasks?

Do Q-learning and C-learning accurately predict the future? Our first experiment studies how well Q-learning and C-learning solve the future state density estimation problem (Def. 2). We use a continuous version of a gridworld for this task and measure how close the predicted future state density is to the true future state density using a KL divergence. Since this environment is continuous and stochastic, Q-learning without hindsight relabelling learns $Q = 0$



(a) Walker2d-v2 (b) Predicted future states

Figure 2.3: **Predicting the Future**: C-learning makes accurate predictions of the expected future state across a range of tasks and discount values. In contrast, learning a 1-step dynamics model and unrolling that model results in high error for large discount values.

on this environment. In the on-policy setting, MC C-learning and TD C-learning perform similarly, while the prediction error for Q-learning (with hindsight relabeling) is more than three times worse (see Appendix H.1 of [57] for details). In the off-policy setting, TD C-learning is more accurate than Q-learning (with hindsight relabeling), achieving a KL divergence that is 14% lower than that of Q-learning. As expected, TD C-learning performs better than MC C-learning in the off-policy setting. In summary, C-learning yields a more accurate solution to the future state density estimation problem, as compared with Q-learning.

Our next experiment studies the ability of C-learning to predict the future in higher-dimensional continuous control tasks. We collected a dataset of experience from agents pre-trained to solve three locomotion tasks from OpenAI Gym. We applied C-learning to each dataset, and used the resulting classifier to predict the expected future state. As a baseline, we trained a 1-step dynamics model on this same dataset and unrolled this model autoregressively to obtain a prediction for the expected future state. Varying the discount factor, we compared each method on `Walker2d-v2` in Fig. 2.3. The 1-step dynamics model is accurate over short horizons but performance degrades for larger values of γ, likely due to prediction errors accumulating over time. In contrast, the predictions obtained by MC C-learning and TD C-learning remain accurate for large values of γ.

Testing our hypotheses about Q-learning: We now test two hypotheses made in Section 2.5.3. The first hypothesis is that Q-learning will underestimate the future state density function. To test this hypothesis, we compute the sum over the predicted future state density function, $\int p^\pi_+(\mathbf{s_{t+}} = s_{t+} \mid \mathbf{s_t}, \mathbf{a_t}) ds_{t+}$, which in theory should equal one. We compared the predictions from MC C-learning and Q-learning using on-policy data. As shown in Fig. 2.1, the predictions from C-learning summed to 1, but the predictions from Q-learning consistently summed to less than one, especially for large values of λ. However, our next experiment shows that Q-learning works best when using large values of λ, suggesting that successful hyperparameters

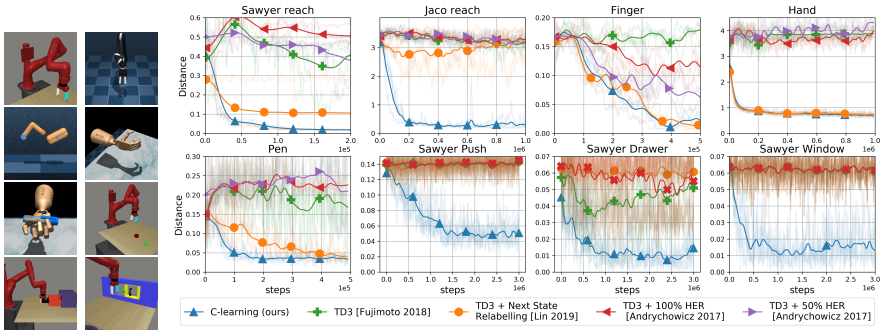


Figure 2.4: **Goal-conditioned RL**: C-learning is competitive with prior goal-conditioned RL methods across a suite of benchmark tasks, without requiring careful tuning of the relabeling distribution. Transparent lines depict individual random seeds.

for Q-learning are ones for which Q-learning does not learn a proper density function.

Our second hypothesis is that Q-learning will perform best when the relabeling ratio is chosen to be $\lambda = (1+\gamma)/2$. As shown in Fig. 2.2, Q-learning is highly sensitive to the relabeling ratio: values of λ that are too large or too small result in Q-learning performing poorly, worse than simply predicting a uniform distribution. Our theoretical hypothesis of $\lambda = (1-\gamma)/2$ almost exactly predicts the optimal value of λ to use for Q-learning. C-learning, which does not depend on this hyperparameter, outperforms Q-learning, even for the best choice of λ. These experiments support our hypothesis for the choice of relabeling ratio while reaffirming that our principled approach to future state density estimation obtains a more accurate solution.

Goal-conditioned RL for continuous control tasks: Our last set of experiments apply goal-conditioned C-learning (Alg. 3) to benchmark continuous control tasks from prior work, shown in Fig. 2.4. These tasks range in difficulty from the 6-dimensional Sawyer Reach task to the 45-dimensional Pen task. The aim of these experiments is to show that C-learning is competitive with prior goal-conditioned RL methods, without requiring careful tuning of the goal sampling ratio. We compare C-learning with a number of prior methods based on Q-learning, which differ in how goals are sampled during training: TD3 [75] does no relabeling, Lin, Baweja, and Held [145] uses 50% next state goals and 50% random goals, and HER [10] uses final state relabeling (we compare against both 100% and 50% relabeling). None of these methods require a reward function or distance function for training; for evaluation, we use the L2 metric between the commanded goal and the terminal state (the average distance to goal and minimum distance to goal show the same trends). As shown in Fig. 2.4, C-learning is competitive with the best of these baselines across all tasks, and substantially better than all baselines on the Sawyer manipulation tasks. These manipulation tasks are more

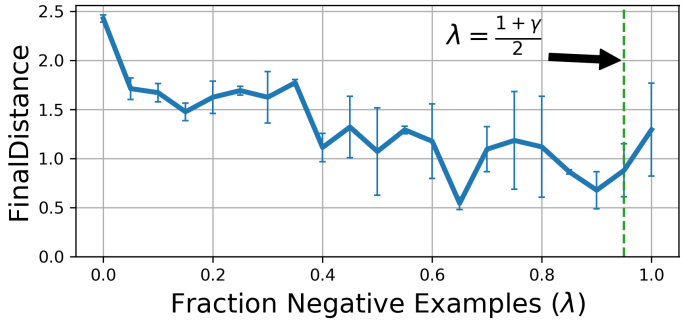


Figure 2.5: Q-learning is sensitive to the relabeling ratio. Our analysis predicts the optimal relabeling ratio.

complex than the others because they require indirect manipulation of objects in the environment. Visualizing the learned policies, we observe that C-learning has discovered regrasping and fine-grained adjustment behaviors, behaviors that typically require complex reward functions to learn [188].[3] On the Sawyer Push and Sawyer Drawer tasks, we found that a hybrid of TD C-learning and MC C-learning performed better than standard C-learning. We describe this variant in Appendix a.3. In summary, C-learning performs as well as prior methods on simpler tasks and better on complex tasks, does not depend on a sensitive hyperparameter (the goal sampling ratio), and maximizes a well-defined objective function.

Goal sampling ratio for goal conditioned RL: While C-learning prescribes a precise method for sampling goals, prior hindsight relabeling methods are sensitive to these parameters. We varied the goal sampling ratio used by Lin, Baweja, and Held [145] on the `maze2d-umaze-v0` task. As shown in Fig. 2.5, properly choosing this ratio can result in a 50% decrease in final distance. Hypothesis 2 provides a good estimate for the best value for this ratio.

2.7 EXTENSION TO FULLY-GENERAL RL PROBLEMS

In subsequent work [51], we extended this idea of recursive classification to fully general RL problems. Instead of specifying tasks via a single goal state, we supposed that a human user provided as input a list of success examples. Intuitively, the method then can use these success examples to reason about what other states might be successful, allowing the method to learn a generalized notion of success. We proved that this framework is fully general, in the sense that any reward maximization task can be represented (up to arbitrary precision) by an appropriately chosen list of success examples. Intuitively, the construction involved including each state with frequency proportional to its reward.

3 See the project website for videos: https://ben-eysenbach.github.io/c_learning

2.8 DISCUSSION

A goal-oriented agent should be able to predict and control the future state of its environment. In this work, we used this idea to reformulate the standard goal-conditioned RL problem as one of estimating and optimizing the future state density function. We showed that Q-learning does not directly solve this problem in (stochastic) environments with continuous states, and hindsight relabeling produces, at best, a mediocre solution for an unclear objective function. In contrast, C-learning yields more accurate solutions. Moreover, our analysis makes two hypotheses about when and where hindsight relabeling will most effectively solve this problem, both of which are validated in our experiments. Our experiments also demonstrate that C-learning scales to high-dimensional continuous controls tasks, where performance is competitive with state-of-the-art goal conditioned RL methods while offering an automatic and principled mechanism for hindsight relabeling.

One limitation of C-learning is that it can struggle to scale to high-dimensional tasks, which are especially important for real-world problems: we expect ML-based decision making tools to be more useful than their hard-coded counterparts precisely because they can cope with large amounts of data. To lift this limitation, the subsequent section extends C-learning by developing a connection between C-learning and contrastive learning. In doing so, we will show that many proposed strategies for learning from unlabeled data (offline RL, representation learning, and model learning) can be seen as emerging from a goal-directed algorithm.

3

CONTRASTIVE LEARNING AS GOAL-CONDITIONED REINFORCEMENT LEARNING

3.1 INTRODUCTION

Representation learning is an integral part of reinforcement learning (RL[1]) algorithms. While such representations might emerge from end-to-end training [11, 143, 227, 247], prior work has found it necessary to equip RL algorithms with perception-specific loss functions [64, 83, 128, 131, 162, 164, 193, 268] or data augmentations [131, 132, 226, 261], 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, 259, 264], it is not surprising that end-to-end learning of representations is fragile [132, 261]: 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 [10, 28, 43, 110, 146, 228], 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 in prior work [177, 210]) while also including actions as an additional input (See Fig. 3.1). 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 effort, 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

1 RL = reinforcement learning, not representation learning.

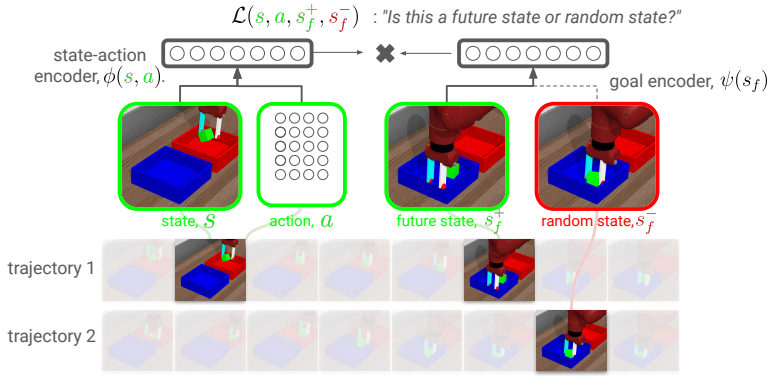


Figure 3.1: **Reinforcement learning via contrastive learning.** Our method uses contrastive learning to acquire representations of state-action pairs ($\phi(s, a)$) and future states ($\psi(s_f)$), so that the representations of future states are closer than the representations of random states. We 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 $\psi(s_g)$.

C-learning [58], 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.

3.2 RELATED WORK

This effort 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 [32, 96, 97, 99, 141, 153, 156, 158, 172, 177, 206, 210, 218, 239, 251, 256]. Contrastive learning methods learn representations such that similar ("positive") examples have similar representations and dissimilar ("negative") examples have dissimilar representations.[2] While most methods generate the "positive" examples via data augmentation, some methods generate similar examples using different camera viewpoints of the same scene [210, 239], or by sampling examples that occur close in time within time series data [7, 177, 210, 226]. 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.

2 Our focus will not be on recent methods that learn representations without negative samples [33, 81].

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 [11, 143, 227, 247], prompting many prior methods to employ separate objectives for representation learning and RL [64, 83, 128, 131, 162, 164, 190, 193, 226, 268, 272]. Many prior methods choose a representation learning objectives that reconstruct the input state [64, 89, 91, 92, 128, 164, 168, 269] while others use contrastive representation learning methods [131, 162, 177, 214, 226]. 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 [27, 38, 65, 72, 111, 119, 166, 257, 258, 276], often in imitation learning settings [71, 98]. 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 effort will focus on goal-conditioned RL problems, a problem prior work has approached using temporal difference learning [10, 58, 110, 146, 198, 202], conditional imitation learning [43, 77, 152, 200, 228], model-based methods [45, 203], hierarchical RL [161], and planning-based methods [56, 168, 200, 222]. The problems of automatically sampling goals and exploration [47, 67, 154, 185, 273] are orthogonal to this work. Like prior work, we will parametrize the value function as an inner product between learned representations [66, 100, 202]. 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 [23, 28, 58, 198] that view goal-conditioned RL as a data-driven problem, rather than as a reward-maximization problem. Many of these methods employ hindsight relabeling [10, 53, 110, 142], 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 effort 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 [58, 110, 164, 245]. Such distance functions are structurally similar to the critic function learned for contrastive learning, a connection we make precisely in Sec. 3.4. In fact, our analysis shows that C-learning [58] 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 [24, 202] and successor features [18, 94, 147]. 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 [23, 255]. While these methods use temporal difference learning, ours is akin to Monte Carlo learning. While Monte Carlo learning is often (but not always [45]) 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 [103, 148, 199, 200, 250]. Our method will learn representations using an objective similar to prior work [200, 210], 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 b.1 for a discussion of how our work relates to unsupervised skill learning.

3.3 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 [9, 84, 237, 254, 265], where tasks correspond to reaching goals states. Following prior work [23, 28, 58, 198], we define the reward as the probability (density) of reaching the goal at the next time step:[3]

$$r_g(s_t, a_t) \triangleq (1-\gamma) p(s_{t+1} = s_g \mid s_t, a_t). \tag{3.1}$$

This reward function is appealing because it avoids the need for a human user to specify a distance metric (unlike, e.g., [10]). 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 define the expected reward objective and Q-function as

$$\max_\pi \mathbb{E}_{p_g(s_g), \pi(\tau|s_g)} \left[\sum_{t=0}^{\infty} \gamma^t r_g(s_t, a_t) \right], \tag{3.2}$$

$$Q^\pi_{s_g}(s, a) \triangleq \mathbb{E}_{\pi(\tau|s_g)} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_g(s_{t'}, a_{t'}) \mid {}^{s_t=s,}_{a_t=a} \right]. \tag{3.3}$$

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

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 [98, 270]

$$p^{\pi(\cdot|\cdot,s_g)}(s_{t+} = s) \triangleq (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p_t^{\pi(\cdot|\cdot,s_g)}(s_t = s), \tag{3.4}$$

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 $\pi(a \mid s) \triangleq \int \pi(a \mid s, s_g) p^\pi(s_g \mid s) ds_g$ [275].

Contrastive representation learning. Contrastive representation learning methods [32, 87, 96, 97, 108, 141, 153, 156, 158, 239, 242, 251] 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 [141, 156, 158, 172]. 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*[4] and note that its range is $(-\infty, \infty)$. We will use NCE-binary [153] objective (also known as InfoMAX [97]):

$$\max_{f(u,v)} \mathbb{E}_{\substack{(u,v^+)\sim p(u,v) \\ v^- \sim p(u)}} \left[\log \sigma(\underbrace{f(u,v^+)}_{\phi(u)^T\psi(v^+)}) + \log(1 - \sigma(\underbrace{f(u,v^-)}_{\phi(u)^T\psi(v^-)})) \right]. \tag{3.5}$$

3.4 CONTRASTIVE LEARNING AS AN RL ALGORITHM

This section shows how to use contrastive representation to *directly* perform goal-conditioned RL. The key idea (Lemma 1) is that con-

4 In contrastive learning, the critic function indicates the similarity between a pair of inputs [187]; in RL, the critic function indicates the future expected returns [118]. Our method combines contrastive learning and RL in a way that these meanings become one and the same.

trastive 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. 3.4.1) and then relate the optimal critic function to the state occupancy measure (Sec. 3.4.2).

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 [58], offering a cogent explanation for its good performance while also suggesting new methods that are simpler and can achieve higher performance.

3.4.1 *Relating the Q-function to probabilities*

This section sets the stage for the main results of this section by providing a probabilistic perspective on goal-conditioned RL. The expected reward objective and associated Q-function in (Eq. 3.3) 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. 3.1) is equivalent to the probability of state s_g under the discounted state occupancy measure:*

$$Q^{\pi}_{s_g}(s, a) = p^{\pi(\cdot|\cdot, s_g)}(s_{t+} = s_g \mid s, a). \tag{3.6}$$

The proof is in Appendix b.3. 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.

3.4.2 *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 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: $s_f \sim p(s_{t+}) \triangleq \int p^{\pi(\cdot|\cdot)}(s_{t+} \mid s, a) p(s, a) ds da$.

With these inputs, the contrastive learning objective (Eq. 3.5) can be written as

$$\max_f \mathbb{E}_{\substack{(s,a)\sim p(s,a), s_f^- \sim p(s_f) \\ s_f^+ \sim p^{\pi(\cdot|\cdot)}(s_{t+}|s_t,a_t)}} \left[\mathcal{L}(s,a,s_f^+, s_f^-) \right],$$

where

$$\mathcal{L}(s,a,s_f^+,s_f^-) \triangleq \log \sigma(\underbrace{f(s,a,s_f^+)}_{\phi(s,a)^T \psi(s_f^+)}) + \log(1 - \sigma(\underbrace{f(s,a,s_f^-)}_{\phi(s,a)^T \psi(s_f^-)})). \quad (3.7)$$

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 [118], figuring out which actions lead to the desired outcome. Because the Bayes-optimal critic function is a function of the state occupancy measure [153], $f^*(s,a,s_g) = \log\left(\frac{p^{\pi(\cdot|\cdot)}(s_{t+}=s_g|s,a)}{p(s_g)}\right)$, it can be used to express the Q-function:

Lemma 1. *The critic function that optimizes Eq. 3.7 is a Q-function for the goal-conditioned reward function (Eq. 3.1), 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 [87], 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. Appendix b.2 discusses how our learned representations can be interpreted as a latent-space model.

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 [43, 77, 152, 228] perform similar sharing, but do not analyze the relationship between the learned policies and Q functions. Sec. 3.4.5 shows that this critic function can be used as the basis for a convergent RL algorithm under some assumptions.

3.4.3 *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:

$$\max_{\pi(a|s,s_g)} \mathbb{E}_{\pi(a|s,s_g)p(s)p(s_g)}\left[f(s,a,s_f = s_g)\right] \approx \mathbb{E}_{\pi(a|s,s_g)p(s)p(s_g)}\left[\log Q_{s_g}^{\pi(\cdot|\cdot)}(s,a) - \log p(s_g)\right]. \tag{3.8}$$

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. 3.4.5, 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. 3.8) 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.

3.4.4 *A Complete Goal-Conditioned RL Algorithm*

The complete algorithm alternates between fitting the critic function using contrastive learning, updating the policy using Eq. 3.8, and collecting more data. Alg. 4 provides a JAX [25] 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 b.4, we derive a variant of this method (`contrastive RL (CPC)`) that uses the infoNCE bound on mutual information.

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 [75, 90]. [5] 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]

5 Code and more results are available: https://ben-eysenbach.github.io/contrastive_rl

6 The more recent DrQ-v2 [260] 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.

Algorithm 4 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
```

3.4.5 *Convergence Guarantees*

In general, providing convergence guarantees for methods that perform relabeling is challenging. Most prior work offers no guarantees [10, 43, 45] or guarantees under only restrictive assumptions [77, 228].

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. b.7 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 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:*

$$\mathbb{E}_{\pi'(\tau|s_g)}\left[\sum_{t=0}^{\infty}\gamma^t r_{s_g}(s_t, a_t)\right] \geq \mathbb{E}_{\pi(\tau|s_g)}\left[\sum_{t=0}^{\infty}\gamma^t r_{s_g}(s_t, a_t)\right] - \frac{2\gamma\epsilon}{1-\gamma}$$

for all goals $s_g \in \{s_g \mid p_g(s_g) > 0\}$.

The proof is in Appendix b.3. This result shows that performing contrastive RL on static dataset results in one step of approximate pol-

icy improvement. Re-collecting data and then applying contrastive RL over and over again corresponds to approximate policy improvement (see [20, 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 [75], target Q networks [159], data augmentation [132, 261], or auxiliary objectives [131, 262].

3.4.6 *C-learning as Contrastive Learning*

C-learning [58] 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.[7] 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 b.5 introduces another member of the contrastive RL family (`contrastive RL (NCE + C-learning)`) that tends to yield the best performance .

3.5 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. Appendix b.6 contain additional experiments and visualizations.

3.5.1 *Comparing to prior goal-conditioned RL methods*

BASELINES. We compare three baselines. "HER" [146] is a goal-conditioned RL method that uses hindsight relabeling [10] with a high-performance actor-critic algorithm (TD3). This baseline is representative of a large class of prior work that uses hindsight relabeling [10, 139, 195, 202]. Like contrastive RL, this baseline does not assume access to a reward function. The second baseline is goal-conditioned behavioral cloning ("GCBC") [30, 43, 50, 77, 152, 182, 225, 228], 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

7 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.

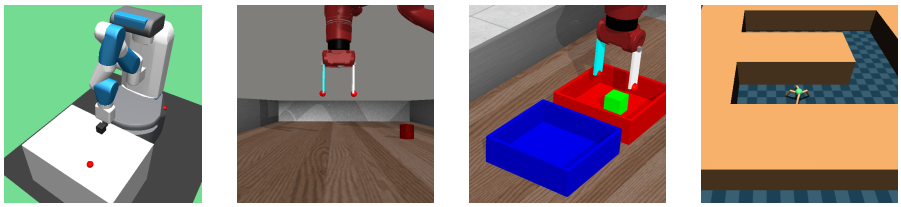
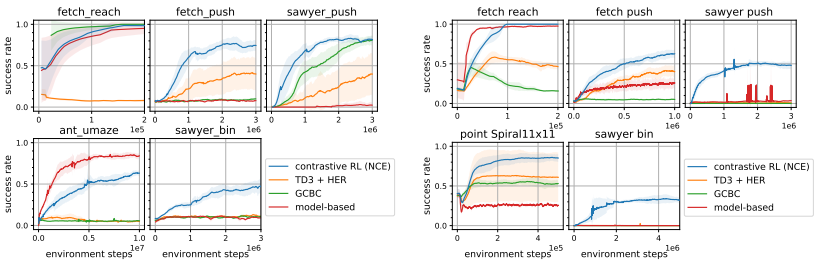


Figure 3.2: **Environments.** We show a subset of the goal-conditioned environments used in our experiments.

results [30, 50] 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 [41] and prior multi-step models [45, 106]. Both contrastive RL (Alg. 4) 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 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. [184] and `sawyer push` and `sawyer bin` from Yu et al. [265]. 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` [70] 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. 3.3, using five random seeds for each experiment and plotting the mean and standard deviation across those random seeds. On the state-based tasks (Fig. 3.3a), 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



(a) State-based tasks (b) Image-based tasks

Figure 3.3: **Goal-conditioned RL.** Contrastive RL (NCE) outperforms prior methods on most tasks. **Baselines**: `HER` [146] is a prototypical actor-critic method that uses hindsight relabeling [10]; `Goal-conditioned behavioral cloning (GCBC)` [43, 77, 152, 225] performs behavior cloning on relabeled experience; `model-based` fits a density model to the discounted state occupancy measure, similar on [41, 45, 106].

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. 3.3b), 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.

3.5.2 *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 [132, 260, 261] ("DrQ") and auxiliary objectives based on an autoencoder [64, 164, 168, 262] ("AE") and a contrastive learning objective ("CURL") that generates positive examples using data augmentation, similar to prior work [131, 162, 226]. 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. 3.4, 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)

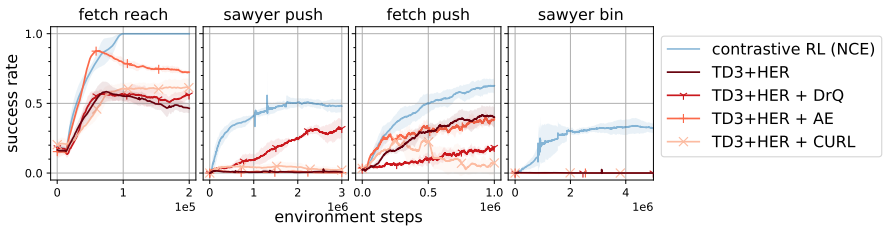


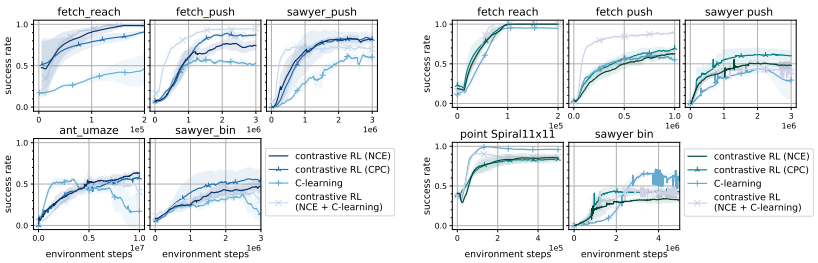
Figure 3.4: **Representation learning for image-based tasks.** While adding data augmentation and auxiliary representation objectives can boost the performance of the `TD3+HER` baseline, replacing the underlying goal-conditioned RL algorithm with one that resembles contrastive representation learning (i.e., ours) yields a larger increase in success rates. **Baselines**: `DrQ` [261] augments images and averages the Q-values across 4 augmentations; `auto encoder (AE)` adds an auxiliary reconstruction loss [64, 164, 168, 262]; `CURL` [131] applies RL on top of representations learned via augmentation-based contrastive learning.

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 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.

3.5.3 *Probing the dimensions of contrastive RL*

Up to now, we have focused on the specific instantiation of contrastive RL spelled out in Alg. 4. 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. 3.4.6). Contrastive RL (CPC) is a variant of Alg. 4 based on the infoNCE objective [177] that we derive in Appendix b.4 Contrastive RL (NCE + C-learning) is a variant that combines C-learning with Alg. b.5 (see Appendix b.5.). 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. 3.5, 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



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

Figure 3.5: **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)).

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 [58], 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.

3.5.4 *Partial Observability and Moving Cameras*

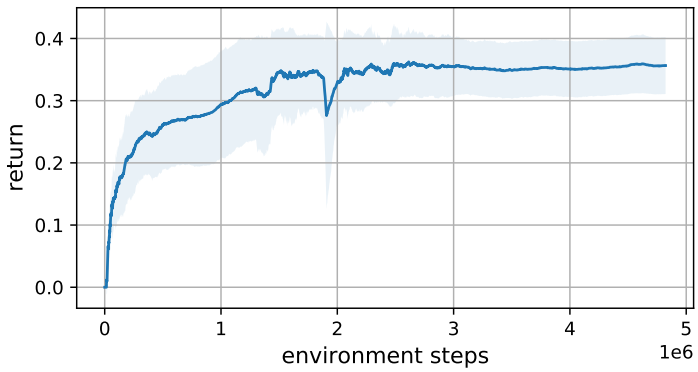


Figure 3.6: **Partial observability and moving cameras.** Contrastive RL can solve partially observed tasks.

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. 3.6 (left)). Nonetheless, contrastive RL (NCE) successfully handles this partial

observability, achieving a success rate of around 35%. Fig. 3.6 (left) shows an example rollout and Fig. 3.6 (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.

3.5.5 *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 [70], 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. 3.8), 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 [30, 43, 50, 77, 152, 182, 225, 228], which we will include as a baseline. Following TD3+BC [73], 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 [50] (which refers to GCBC as RvS-G), and a recent method based on the transformer architecture (DT [30]). Lastly, we compare with two more complex methods that use TD learning: TD3+BC [73] and IQL [120]. 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 [50, 120].

As shown in Table 3.1, 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.

Table 3.1: **Offline RL on D4RL AntMaze** [70]. Contrastive RL outperforms all baselines in 5 out of 6 tasks. TD3+BC and IQL report results on the `-v0` tasks, but the change to `-v2` has a negligible effect on TD methods [102].

	no TD					uses TD	
	BC	DT	GCBC	Contrastive RL + BC		TD3+BC*	IQL*
				2 nets	5 nets		
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

3.6 EXTENSIONSION: SOLVING FULLY-GENERAL RL PROBLEMS USING CONTRASTIVE KERNELS

While contrastive RL focuses on goal-reaching problems, the underlying mathematics can allow us to solve fully general RL problems. In particularly, these contrastive representations provide an immediate solution to the example-based control problem mentioned in Sec. 2.7. Precisely, estimating the Q function corresponds to kernel smoothing, where the Q function is represented non-parametrically as a reward-weighted sum of the distances to reward-labeled states (according to the representations) [95].

3.7 DISCUSSION

In this work, 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 work might be construed to imply that RL is more or less important than representation learning [129, 134, 216, 223], we have a different takeaway: that it may be enough to build RL algorithms that *look like* representation learning. Moreoever, this work reveals close connections between various unsupervised pretraining methods (representation learning, model learning, offline RL, and goal-conditioned RL), suggesting that they may largely be different interpretation of the same method.

In summary, both C-learning and its contrastive brethren have lifted both practical and theoretical limitations of prior work. On the practical front, unlike prior methods, they do not require manual specification of reward functions or distance functions, and they

automatically suggest how relabeling hyperparameters should be chosen. They also highlight how representation can emerge from goal-conditioned RL. On the theoretical front, they show how hindsight relabeling can be used in a principled manner, resulting in well-defined value functions. Taken together, these methods underscore how data-driven decision making algorithms are not only practically appealing, but also open the door to theoretical inquiry.

Part II

INFERRING SOLUTIONS TO COMPLEX TASKS

Directly solving difficult tasks, without any sort of plan, can be exceedingly difficult. This section discusses RL algorithms that infer a plan for solving a task. Like the scaffolding of a building, this plan makes the subsequent control problem easier, as the learned agent need only fill in the gaps in the scaffolding. Different choices of scaffolding result in different control algorithms. Chapter 4 uses a coarse scaffold that consists of a sequence of waypoints leading to a desired outcome. Chapter 5 uses a fine-grained scaffold that models the dynamics at every time step.

4

SEARCH ON THE REPLAY BUFFER

4.1 INTRODUCTION

How can agents learn to solve complex, temporally extended tasks? Classically, planning algorithms give us one tool for learning such tasks. While planning algorithms work well for tasks where it is easy to determine distances between states and easy to design a local policy to reach nearby states, both of these requirements become roadblocks when applying planning to high-dimensional (e.g., image-based) tasks. Learning algorithms excel at handling high-dimensional observations, but reinforcement learning (RL) – learning for control – fails to reason over long horizons to solve temporally extended tasks. In this work, we propose a method that combines the strengths of planning and RL, resulting in an algorithm that can plan over long horizons in tasks with high-dimensional observations.

Recent work has introduced goal-conditioned RL algorithms [186, 202] that acquire a single policy for reaching many goals. In practice, goal-conditioned RL succeeds at reaching nearby goals but fails to reach distant goals; performance degrades quickly as the number of steps to the goal increases [140, 161]. Moreover, goal-conditioned RL often requires large amounts of reward shaping [34] or human demonstrations [151, 165], both of which can limit the asymptotic performance of the policy by discouraging the policy from seeking novel solutions.

We propose to solve long-horizon, sparse reward tasks by decomposing the task into a series of easier goal-reaching tasks. We learn a goal-conditioned policy for solving each of the goal-reaching tasks. Our main idea is to reduce the problem of finding these subgoals to solving a shortest path problem over states that we have previous visited, using a distance metric extracted from our goal-conditioned policy. We call this algorithm Search on Replay Buffer (SoRB), and provide a simple illustration of the algorithm in Figure 4.1.

Our primary contribution is an algorithm that bridges planning and deep RL for solving long-horizon, sparse reward tasks. We develop a practical instantiation of this algorithm using ensembles of distributional value functions, which allows us to *robustly* learn distances and use them for *risk-aware* planning. Empirically, we find that our method generates effective plans to solve long horizon navigation tasks, even in image-based domains, without a map and without odometry. Comparisons with state-of-the-art RL methods show that SoRB is substantially more successful in reaching distant goals. We also

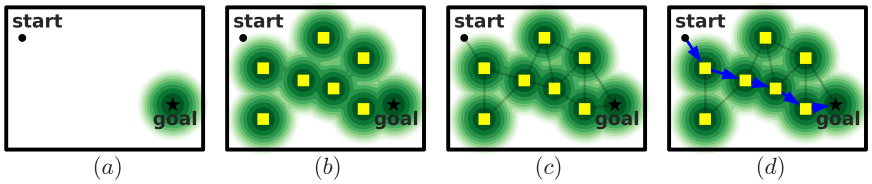


Figure 4.1: **Search on the Replay Buffer:** (a) Goal-conditioned RL often fails to reach distant goals, but can successfully reach the goal if starting nearby (inside the green region). (b) Our goal is to use observations in our replay buffer (yellow squares) as waypoints leading to the goal. (c) We automatically find these waypoints by using the agent's value function to predict when two states are nearby, and building the corresponding graph. (d) We run graph search to find the sequence of waypoints (blue arrows), and then use our goal-conditioned policy to reach each waypoint.

observe that the learned policy generalizes well to navigate in unseen environments. In summary, graph search over previously visited states is a simple tool for boosting the performance of a goal-conditioned RL algorithm.

4.2 BRIDGING PLANNING AND REINFORCEMENT LEARNING

Planning algorithms must be able to (1) sample valid states, (2) estimate the distance between reachable pairs of states, and (3) use a local policy to navigate between nearby states. These requirements are difficult to satisfy in complex tasks with high dimensional observations, such as images. For example, consider a robot arm stacking blocks using image observations. Sampling states requires generating photo-realistic images, and estimating distances and choosing actions requires reasoning about dozens of interactions between blocks. Our method will obtain distance estimates and a local policy using a RL algorithm. To sample states, we will simply use a replay buffer of previously visited states as a non-parametric generative model.

4.2.1 *Building Block: Goal-Conditioned RL*

A key building block of our method is a goal-conditioned policy and its associated value function. We consider a goal-reaching agent interacting with an environment. The agent observes its current state $s \in \mathcal{S}$ and a goal state $s_g \in \mathcal{S}$. The initial state for each episode is sampled $s_1 \sim \rho(s)$, and dynamics are governed by the distribution $p(s_{t+1} \mid s_t, a_t)$. At every step, the agent samples an action $a \sim \pi(a \mid s, s_g)$ and receives a corresponding reward $r(s, a, s_g)$ that indicates whether the agent has reached the goal. The episode terminates as soon as the agent reaches the goal, or after T steps, whichever occurs first. The agent's task is to maximize its cumulative, *undiscounted*,

reward. We use an off-policy algorithm to learn such a policy, as well as its associated goal-conditioned Q-function and value function:

$$Q(s, a, s_g) = \mathbb{E}_{\substack{s_1 \sim \rho(s), a_t \sim \pi(a_t | s_t, s_g) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_{t=1}^{T} r(s_t, a_t, s_g) \right],$$
$$V(s, s_g) = \max_a Q(s, a, s_g).$$

We obtain a policy by acting greedily w.r.t. the Q-function: $\pi(a \mid s, s_g) = \arg\max_a Q(s, a, s_g)$. We choose an off-policy RL algorithm with goal relabelling [10, 110] and distributional RL [19]) not only for improved data efficiency, but also to obtain good distance estimates (See Section 4.2.2). We will use DQN [159] for discrete action environments and DDPG [144] for continuous action environments. Both algorithms operate by minimizing the Bellman error over transitions sampled from a replay buffer $\mathcal{B}$.

4.2.2 *Distances from Goal-Conditioned Reinforcement Learning*

To ultimately perform planning, we need to compute the *shortest path distance* between pairs of states. Following Kaelbling [110], we define a reward function that returns -1 at every step: $r(s, a, s_g) \triangleq -1$. The episode ends when the agent is sufficiently close to the goal, as determined by a state-identity oracle. Using this reward function and termination condition, there is a close connection between the Q values and shortest paths. We define $d_{\text{sp}}(s, s_g)$ to be the shortest path distance from state s to state s_g. That is, $d_{\text{sp}}(s, s_g)$ is the expected number of steps to reach s_g from s under the optimal policy. The value of state s with respect to goal s_g is simply the negative shortest path distance: $V(s, s_g) = -d_{\text{sp}}(s, s_g)$. We likewise define $d_{\text{sp}}(s, a, s_g)$ as the shortest path distance, conditioned on initially taking action a. Then Q values also equal a negative shortest path distance: $Q(s, a, s_g) = -d_{\text{sp}}(s, a, s_g)$. Thus, goal-conditioned RL on a suitable reward function yields a Q-function that allows us to estimate shortest-path distances.

4.2.3 *The Replay Buffer as a Graph*

We build a weighted, *directed* graph directly on top of states in our replay buffer, so each node corresponds to an observation (e.g., an image). We add edges between nodes with weight (i.e., length) equal to their predicted distance, using $d_\pi(s_1, s_2)$ as our estimate of the distance using our current Q-function. While, in theory, going directly to the goal is always a shortest path, in practice the goal-conditioned policy will fail to reach distant goals directly (See Fig. 4.5.). We will therefore

ignore edges that are longer than MaxDist, a hyperparameter. The graph is thus defined as $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E}, \mathcal{W})$ where

$$\mathcal{V} = \mathcal{B}, \quad \mathcal{E} = \mathcal{B} \times \mathcal{B} = \{e_{s_1 \to s_2} \mid s_1, s_2 \in \mathcal{B}\}$$

$$\mathcal{W}(e_{s_1 \to s_2}) = \begin{cases} d_\pi(s_1, s_2) & \text{if } d_\pi(s_1, s_2) < \text{MaxDist} \\ \infty & \text{otherwise.} \end{cases}$$

Given a start and goal state, we temporarily add each to the graph. We add directed edges from the start state to every other state, and from every other state to the goal state, using the same criteria as above. We use Dijkstra's Algorithm to find the shortest path. See Appendix c for details.

Algorithm 5 Inputs are the current state s, the goal state s_g, a buffer of observations $\mathcal{B}$, the learned policy π and its value function V. Returns an action a.

function SearchPolicy($s, s_g, \mathcal{B}, V, \pi$)
  $s_{w_1}, \cdots \leftarrow$ ShortestPath($s, s_g, \mathcal{B}, V$)
  $d_{s \to w_1} \leftarrow -V(s, s_{w_1})$
  $d_{s \to g} \leftarrow -V(s, s_g)$
  if $d_{s \to w_1} < d_{s \to g}$ or $d_{s \to g} >$ MaxDist **then**
    $a \leftarrow \pi(a, \mid s, s_{w_1})$
  else
    $a \leftarrow \pi(a, \mid s, s_g)$
  return a

4.2.4 *Algorithm Summary*

After learning a goal-conditioned Q-function, we perform graph search to find a set of waypoints and use the goal-conditioned policy to reach each. We view the combination of graph search and the underlying goal-conditioned policy as a new SearchPolicy, shown in Algorithm 5. The algorithm starts by using graph search to obtain the shortest path $s_{w_1}, s_{w_2}, \cdots$ from the current state s to the goal state s_g, planning over the states in our replay buffer $\mathcal{B}$. We then estimate the distance from the current state to the first waypoint, as well as the distance from the current state to the goal. In most cases, we then condition the policy on the first waypoint, s_{w_1}. However, if the goal state is closer than the next waypoint and the goal state is not too far away, then we directly condition the policy on the final goal. If the replay buffer is empty or there is not a path in $\mathcal{G}$ to the goal, then Algorithm 5 resorts to standard goal-conditioned RL.

4.3 BETTER DISTANCE ESTIMATES

The success of our SEARCHPOLICY depends heavily on the accuracy of our distance estimates. This section proposes two techniques to learn better distances with RL.

4.3.1 *Better Distances via Distributional Reinforcement Learning*

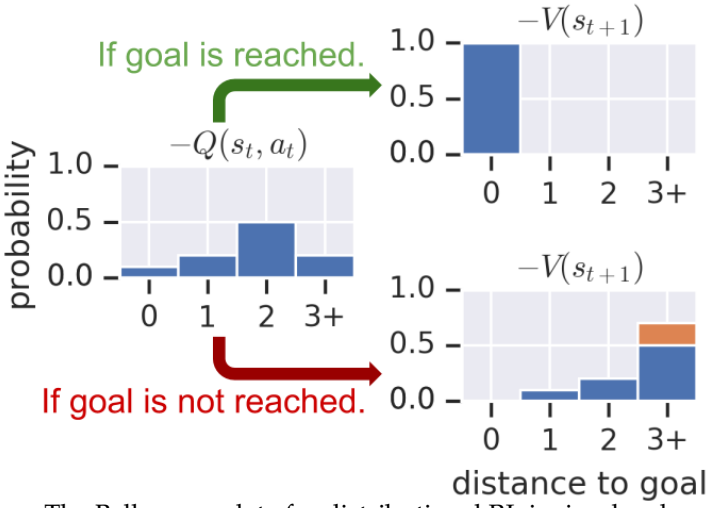


Figure 4.2: The Bellman update for distributional RL is simple when learning distances, simply corresponding to a left-shift of the Q-values at every step until the agent reaches the goal.

Off-the-shelf Q-learning algorithms such as DQN [159] or DDPG [144] will fail to learn accurate distance estimates using the -1 reward function. The true value for a state and goal that are unreachable is $-\infty$, which cannot be represented by a standard, feed-forward Q-network. Simply clipping the Q-value estimates to be within some range avoids the problem of ill-defined Q-values, but empirically we found it challenging to train clipped Q-networks. We adopt distributional Q-learning [19], noting that is has a convenient form when used with the -1 reward function. Distributional RL discretizes the possible value estimates into a set of bins $B = (B_1, B_2, \cdots, B_N)$. For learning distances, bins correspond to distances, so B_i indicates the event that the current state and goal are i steps away from one another. Our Q-function predicts a distribution $Q(s_t, s_g, a_t) \in \mathcal{P}^N$ over these bins, where $Q(s_t, s_g, a_t)_i$ is the predicted probability that states s_t and s_g are i steps away from one another. To avoid ill-defined Q-values, the final bin, B_N is a catch-all for predicted distances of at least N. Importantly, this gives us a well-defined method to represent large

and infinite distances. Under this formulation, the targets $Q^* \in \mathcal{P}^N$ for our Q-values have a simple form:

$$Q^* = \begin{cases} (1, 0, \cdots, 0) & \text{if } s_t = g \\ (0, Q_1, \cdots, Q_{N-2}, Q_{N-1} + Q_N) & \text{if } s_t \neq g \end{cases}$$

As illustrated in Figure 4.2, if the state and goal are equivalent, then the target places all probability mass in bin 0. Otherwise, the targets are a right-shift of the current predictions. To ensure the target values sum to one, the mass in bin N of the targets is the sum of bins $N-1$ and N from the predicted values. Following Bellemare, Dabney, and Munos [19], we update our Q function by minimizing the KL divergence between our predictions Q^θ and the target Q^*:

$$\min_\theta D_{\text{KL}}(Q^* \parallel Q^\theta) \tag{4.1}$$

4.3.2 *Robust Distances via Ensembles of Value Functions*

Since we ultimately want to use estimated distances to perform search, it is crucial that we have accurate distances estimates. It is challenging to robustly estimate the distance between all $|\mathcal{B}|^2$ pairs of states in our buffer $\mathcal{B}$, some of which may not have occurred during training. If we fail and spuriously predict that a pair of distant states are nearby, graph search will exploit this "wormhole" and yield a path which assumes that the agent can "teleport" from one distant state to another. We seek to use a bootstrap [21] as a principled way to estimate uncertainty for our Q-values. Following prior work [126, 179], we implement an approximation to the bootstrap. We train an ensemble of Q-networks, each with independent weights, but trained on the same data using the same loss (Eq. 4.1). When performing graph search, we aggregate predictions from each Q-network in our ensemble. Empirically, we found that ensembles were crucial for getting graph search to work on image-based tasks, but we observed little difference in whether we took the maximum predicted distance or the average predicted distance.

4.4 RELATED WORK

Planning Algorithms: Planning algorithms [36, 125] efficiently solve long-horizon tasks, including those that stymie RL algorithms (see, e.g., Kavraki, Svestka, and Overmars [113], Lau and Kuffner [133], and Levine et al. [138]). However, these techniques assume that we can (1) efficiently sample valid states, (2) estimate the distance between two states, and (3) acquire a local policy for reaching nearby states, all of which make it challenging to apply these techniques to high-dimensional tasks (e.g., with image observations). Our method re-

moves these assumptions by (1) sampling states from the replay buffer and (2,3) learning the distance metric and policy with RL. Some prior works have also combined planning algorithms with RL [34, 62, 200], finding that the combination yields agents adept at reaching distant goals. Perhaps the most similar work is Semi-Parametric Topological Memory [200], which also uses graph search to find waypoints for a learned policy. We compare to SPTM in Section 4.5.3.

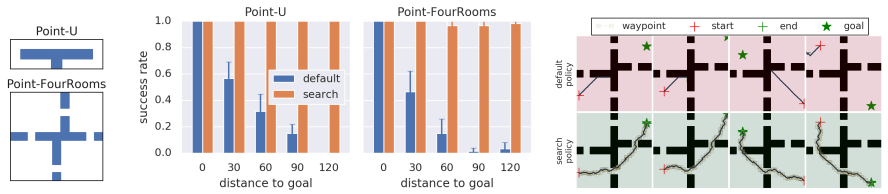
Goal-Conditioned RL: Goal-conditioned policies [110, 186, 202] take as input the current state and a goal state, and predict a sequence of actions to arrive at the goal. Our algorithm learns a goal-conditioned policy to reach waypoints along the planned path. Recent algorithms [10, 186] combine off-policy RL algorithms with goal-relabelling to improve the sample complexity and robustness of goal-conditioned policies. Similar algorithms have been proposed for visual navigation [8, 86, 157, 267, 274]. A common theme in recent work is learning distance metrics to accelerate RL. While most methods [66, 201, 255] simply perform RL on top of the learned representation, our method explicitly performs search using the learned metric.

Hierarchical RL: Hierarchical RL algorithms automatically learn a set of primitive skills to help an agent learn complex tasks. One class of methods [14, 69, 109, 121, 161, 181, 189, 231, 246] jointly learn a low-level policy for performing each of the skills together with a high-level policy for sequencing these skills to complete a desired task. Another class of algorithms [46, 68, 217] focus solely on automatically discovering these skills or subgoals. SoRB learns primitive skills that correspond to goal-reaching tasks, similar to Nachum et al. [161]. While jointly learning high-level and low-level policies can be unstable (see discussion in Nachum et al. [161]), we sidestep the problem by using graph search as a fixed, high-level policy.

Table 4.1: Four classes of model-based RL methods. Dimensions in the last column correspond to typical robotics tasks with image/lidar observations.

model	real states	multi-step	prediction dimension
state-space	✓	✓	1000s+
latent-space	✗	✓	10s
inverse	✓	✗	10s
SoRB	✓	✓	1

Model Based RL: RL methods are typically divided into model-free [207–209, 253] and model-based [144, 249] approaches. Model-based approaches all perform some degree of planning, from predicting the value of some state [159, 215], obtaining representations by unrolling a learned dynamics model [191], or learning a policy directly



(a) (b) (c)

Figure 4.3: **Simple 2D Navigation**: *(Left)* Two simple navigation environments. *(Center)* An agent that combines a goal-conditioned policy with search is substantially more successful at reaching distant goals in these environments than using the goal-conditioned policy alone. *(Right)* A standard goal-conditioned policy (top) fails to reach distant goals. Applying graph search on top of that *same policy* (bottom) yields a sequence of intermediate waypoints (yellow squares) that enable the agent to successfully reach distant goals.

on a learned dynamics model [3, 39, 63, 123, 163, 173, 230]. One line of work [6, 135, 224, 235] embeds a differentiable planner inside a policy, with the planner learned end-to-end with the rest of the policy. Other work [136, 250] explicitly learns a representation for use inside a standard planning algorithm. In contrast, SoRB learns to predict the distances between states, which can be viewed as a high-level inverse model. SoRB predicts a scalar (the distance) rather than actions or observations, making the prediction problem substantially easier. By planning over previously visited states, SoRB does not have to cope with infeasible states that can be predicted by forward models in state-space and latent-space.

4.5 EXPERIMENTS

We compare SoRB to prior methods on two tasks: a simple 2D environment, and then a visual navigation task, where our method will plan over images. Ablation experiments will illustrate that accurate distances estimates are crucial to our algorithm's success.

4.5.1 *Didactic Example: 2D Navigation*

We start by building intuition for our method by applying it to two simple 2D navigation tasks, shown in Figure 4.3a. The start and goal state are chosen randomly in free space, and reaching the goal often takes over 100 steps, even for the optimal policy. We used goal-conditioned RL to learn a policy for each environment, and then evaluated this policy on randomly sampled (start, goal) pairs of varying difficulty. To implement SoRB, we used exactly the same policy, both to perform graph search and then to reach each of the planned waypoints. In Figure 4.3b, we observe that the goal-conditioned policy

can reach nearby goals, but fails to generalize to distant goals. In contrast, SoRB successfully reaches goals over 100 steps away, with little drop in success rate. Figure 4.3c compares rollouts from the goal-conditioned policy and our policy. Note that our policy takes actions that temporarily lead away from the goal so the agent can maneuver through a hallway to eventually reach the goal.

4.5.2 *Planning over Images for Visual Navigation*

Figure 4.4: **Visual Navigation**: Given an initial state and goal state, our method automatically finds a sequence of intermediate waypoints. The agent then follows those waypoints to reach the goal.

We now examine how our method scales to high-dimensional observations in a visual navigation task, illustrated in Figure 4.4. We use 3D houses from the SUNCG dataset [220], similar to the task described by Shah et al. [212]. The agent receives either RGB or depth images and takes actions to move North/South/East/West. Following Shah et al. [212], we stitch four images into a panorama, so the resulting observation has dimension $4 \times 24 \times 32 \times C$, where C is the number of channels (3 for RGB, 1 for Depth). At the start of each episode, we randomly sample an initial state and goal state. We found that sampling nearby goals (within 4 steps) more often (80% of the time) improved the performance of goal-conditioned RL. We use the same goal sampling distribution for all methods. The agent observes both the current image and the goal image, and should take actions that lead to the goal state. The episode terminates once the agent is within 1 meter of the goal. We also terminate if the agent has failed to reach the goal after 20 time steps, but treat the two types of termination differently when computing the TD error (see Pardo et al. [180]). Note that it is challenging to specify a meaningful distance metric and local policy on pixel inputs, so it is difficult to apply standard planning algorithms to this task.

On this task, we evaluate four state-of-the-art prior methods: hindsight experience replay (HER) [10], distributional RL (C51) [19], semi-parametric topological memory (SPTM) [200], and value iteration net-

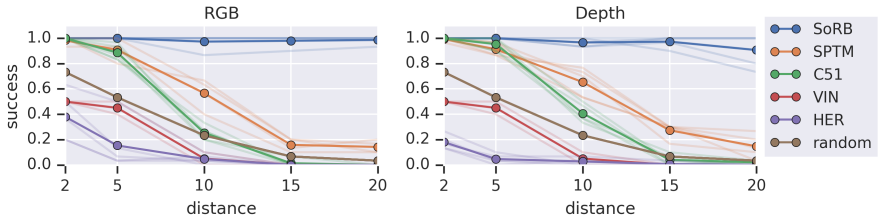


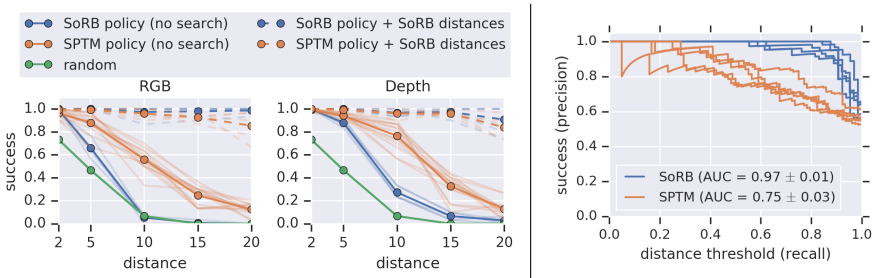
Figure 4.5: **Visual Navigation**: We compare our method (SoRB) to prior work on the visual navigation environment (Fig. 4.4), using RGB images *(Left)* and depth images *(Right)* . We find that only our method succeeds in reaching distant goals. *Baselines*: SPTM [200], C51 [19], VIN [235], HER [10]. Transparent lines depict individual random seeds.

works (VIN) [235]. SoRB uses C51 as its underlying goal-conditioned policy. For VIN, we tuned the number of iterations as well as the number of hidden units in the recurrent layer. For SPTM, we performed a grid search over the threshold for adding edges, the threshold for choosing the next waypoint along the shortest path, and the parameters for sampling the training data. In total, we performed over 1000 experiments to tune baselines, more than an order of magnitude more than we used for tuning our own method. See Appendix C for details.

We evaluated each method on goals ranging from 2 to 20 steps from the start. For each distance, we randomly sampled 30 (start, goal) pairs, and recorded the average success rate, defined as reaching within 1 meter of the goal within 100 steps. We then repeated each experiment for 5 random seeds. In Figure 4.5, we plot each random seed as a transparent line; the solid line corresponds to the average across the 5 random seeds. While all prior methods degrade quickly as the distance to the goal increases, our method continues to succeed in reaching goals with probability around 90%. SPTM, the only prior method that also employs search, performs second best, but substantially worse than our method.

4.5.3 *Comparison with Semi-Parametric Topological Memory*

To understand why SoRB succeeds at reaching distant goals more frequently than SPTM, we examine the two key differences between the methods: (1) the *goal-conditioned policy* used to reach nearby goals and (2) the *distance metric* used to construct the graph. While SoRB acquires a goal-conditioned policy via goal-conditioned RL, SPTM obtains a policy by learning an inverse model with supervised learning. First, we compared the performance of the RL policy (used in SoRB) with the inverse model policy (used in SPTM). In Figure 4.6a, the solid colored lines show that, *without search*, the policy used by SPTM is more successful than the RL policy, but performance of both policies



(a) Goal-Conditioned Policy

(b) Distance Predictions

Figure 4.6: **SoRB vs SPTM**: Our method and Semi-Parametric Topological Memory [201] differ in the policy used and how distances are estimated. We find *(Left)* that both methods learn comparable policies, but *(Right)* our method learns more accurate distances. Transparent lines depict individual random seeds.

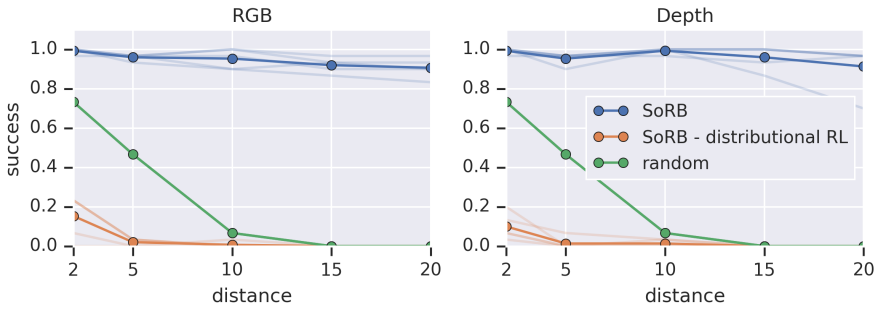
degrades as the distance to the goal increases. We also evaluate a variant of our method that uses the policy from SPTM to reach each waypoint, and find (dashed-lines) no difference in performance, likely because the policies are equally good at reaching nearby goals (within MaxDist steps). We conclude that the difference in goal-conditioned policies cannot explain the difference in success rate.

The other key difference between SoRB and SPTM is their learned distance metrics. When using distances for graph search, it is critical for the predicted distance between two states to reflect whether the policy can successfully navigate between those states: the model should be more successful at reaching goals which it predicts are nearby. We can naturally measure this alignment using the area under a precision recall curve. Note that while SoRB predicts distances in the range $[0, T]$, SPTM predicts whether two states are reachable, so its predictions will be in the range $[0, 1]$. Nonetheless, precision-recall curves[1] only depend on the ordering of the predictions, not their absolute values. Figure 4.6b shows that the distances predicted by SoRB more accurately reflect whether the policy will reach the goal, as compared with SPTM. The average AUC across five random seeds is 22% higher for SoRB than SPTM. In retrospect, this finding is not surprising: while SPTM employs a learned, inverse model policy, it learns distances w.r.t. a random policy.

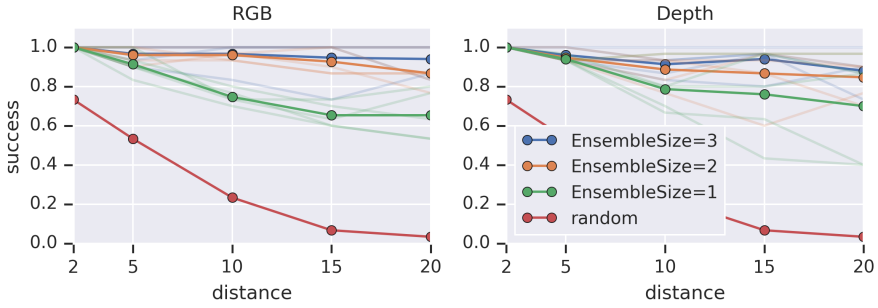
4.5.4 *Better Distance Estimates*

We now examine the ingredients in SoRB that contribute to its accurate distance estimates: distributional RL and ensembles of value functions. In a first experiment, evaluated a variant of SoRB trained without distributional RL. As shown in Figure 4.7a, this variant performed

1 We negate the distance prediction from SoRB before computing the precision recall curve because small distances indicate that the policy should be more successful.



(a) Distributional RL



(b) Ensembles

Figure 4.7: **Better Distance Estimates**: *(Left)* Without distributional RL, our method performs poorly. *(Right)* Ensembles contribute to a moderate increase in success rate, especially for distant goals.

worse than the random policy, clearly illustrating that distributional RL is a key component of SoRB. The second experiment studied the effect of using ensembles of value functions. Recalling that we introduced ensembles to avoid erroneous distance predictions for distant pairs of states, we expect that ensembles will contribute most towards success at reaching distant goals. Figure 4.7b confirms this prediction, illustrating that ensembles provide a 10 – 20% increase in success at reaching goals that are at least 10 steps away. We run additional ablation analysis in Appendix c.

4.5.5 *Generalizing to New Houses*

We now study whether our method generalizes to new visual navigation environments. We train on 100 SUNCG houses, randomly sampling one per episode. We evaluated on a held-out test set of 22 SUNCG houses. In each house, we collect 1000 random observations and fill our replay buffer with those observations to perform search. We use the same goal-conditioned policy and associated distance function that we learned during training. As before, we measure the fraction of goals reached as we increase the distance to the goal. In Figure 4.8, we observe that SoRB reaches almost 80% of goals that

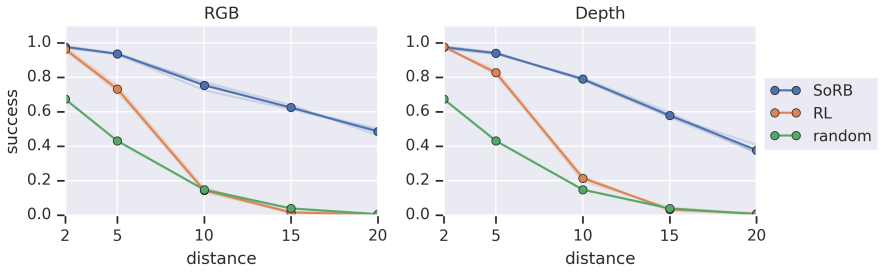


Figure 4.8: **Does SoRB Generalize?** After training on 100 SUNCG houses, we collect random data in held-out houses to use for search in those new environments. Whether using depth images or RGB images, SoRB generalizes well to new houses, reaching almost 80% of goals 10 steps away, while goal-conditioned RL reaches less than 20% of these goals. Transparent lines correspond to average success rate across 22 held-out houses for each of three random seeds.

are 10 steps away, about four times more than reached by the goal-conditioned RL agent. Our method succeeds in reaching 40% of goals 20 steps away, while goal-conditioned RL has a success rate near 0%. We repeated the experiment for three random seeds, retraining the policy from scratch each time. Note that there is no discernible difference between the three random seeds, plotted as transparent lines, indicating the robustness of our method to random initialization.

4.6 EXTENSIONS

On the empirical side, we worked together with collaborators to develop a version of SoRB that could drive a small robot (Fig. 4.9) around a small outdoor campus, given only an image of the desired location [211]. We demonstrated applications included contact-free pizza delivery and autonomous mail delivery.

Figure 4.9: Developing a variant of SoRB for real-world outdoor visual navigation [211].

While the SoRB algorithm worked well in practice, from a theoretical perspective there remained a few open questions. *First*, SoRB requires an ϵ-ball for defining whether the agent has reached the goal state or not. Specifying such balls requires some intuition about what metrics make sense for the given task,

and require tuning a hyperparameter ϵ. *Second*, SoRB does not take into account the distribution of the waypoints; if the replay buffer contains a non-uniform distribution of observations, then the graph will be biased towards containing certain types of observations, and the planning procedure does not take that distribution into account.

In subsequent work [271], we lift these two limitations by providing a probabilistic treatment of SoRB. This required using C-learning (Chapter 2) to learn the edges of the graph. In the end, we can prove that this new method achieve even better empirical results while retaining stronger theoretical guarantees.

4.7 DISCUSSION

We presented SoRB, a method that combines planning via graph search and goal-conditioned RL. By exploiting the structure of goal-reaching tasks, we can obtain policies that generalize substantially better than those learned directly from RL. In our experiments, we show that SoRB can solve temporally extended navigation problems, traverse environments with image observations, and generalize to new houses in the SUNCG dataset. Broadly, we expect SoRB to outperform existing RL approaches on long-horizon tasks, especially those with high-dimensional inputs. In addition, while the planning algorithm we use is simple (namely, Dijkstra), we believe that the key idea of using distance estimates obtained from RL algorithms for planning will open doors to incorporating more sophisticated planning techniques into RL.

One way of interpretating SoRB (see [271]) is that it infers a *coarse* set of waypoints leading from one state to another. In the next chapter we describe an alternative parametrizatoin, where we attempt to infer the *dense* sequence of observations leading to high-reward states.

5

JOINT MODEL-POLICY OPTIMIZATION FOR MODEL-BASED RL

5.1 INTRODUCTION

Much of the appeal of model-based RL is that model learning is a simple and scalable supervised learning problem. Unfortunately, the accuracy of the learned model does not directly correlate with whether the model-based RL algorithm will receive high reward [61, 127]. For example, a model might make small mistakes in critical states that cause a policy to take suboptimal actions. Alternatively, a model with large errors may yield a policy that attains high return if the model errors occur in states that the policy never visits.

The underlying problem is that dynamics models are trained differently from how they are used. Typical model-based methods train a model using data sampled from the *real* dynamics (e.g., using maximum likelihood), but apply these models by using data sampled from the *learned* dynamics [42, 91, 105, 252]. Prior work has identified this *objective mismatch* issue [61, 127, 150]: the model is trained using one objective, but the policy is trained using a different objective. Designing an objective for model training that is guaranteed to improve the expected reward remains an open problem. This work aims to answer the following question: *How should we train a dynamics model so that it produces high-return policies when used for model-based RL?*

In this work, we propose a model-based RL algorithm where the model and policy are *jointly optimized* with respect to the same objective. Our objective is a lower bound on the expected return under the true environment dynamics; a slightly more complicated version of this lower bound becomes tight under certain assumptions. Structurally, our algorithm resembles a generative adversarial network (a GAN), in that the model is trained using a discriminator that distinguishes between real and fake transitions. This same discriminator is included in the objective for the policy, and both the model and policy are jointly trained to maximize reward and minimize discriminator accuracy. Thus, the model and policy *cooperate* to produce realistic and high-reward trajectories. Our method stands in contrast to standard model-based RL methods, where it is more common to pit the model *against* the policy [15, 171, 197]. An consequence of maximizing the lower bound is that the dynamics model does not learn the true dynamics, but rather learns optimistic dynamics that facilitate exploration.

The main contribution of this work is an algorithm, Mismatched no More (MnM), for model-based RL that provably maximizes a lower bound on expected reward. Importantly, this bound becomes tight at optimality under certain assumptions. To the best of our knowledge, this is the first model-based RL objective that is a global lower bound on expected return, and that involves optimizing the model and policy using the same objective. Our algorithm has the unique property of jointly optimizing the policy and model using the same objective. Across a range of tasks, we demonstrate that our method is competitive with prior state-of-the-art methods on benchmark tasks; on certain hard exploration tasks, our method outperforms prior methods based on maximum likelihood estimation.

5.2 RELATED WORK

Model-based RL methods typically fit a dynamics model to observed transitions, and then apply an RL method to that learned model. Most of these methods use maximum likelihood to fit the dynamics model, and then use RL to maximize the expected return under samples from that model [39, 42, 91, 105, 252]. The observation that this maximum likelihood objective is not aligned with the RL objective has been noted in prior work [61, 127, 150, 234, 275]. This issue is referred to as the *objective mismatch* problem: the model and policy (or planner) are optimized using different objectives. This problem arises in almost all model-based RL approaches, including those that train the model to predict the value function [175, 204] or that perform planning [39, 204].

One strategy for mitigating this problem is to modify the model training to improve model accuracy under multi-step rollouts [12, 13, 61, 107, 234, 244]. A second strategy is penalize the policy for taking transitions where the model is inaccurate [115, 150, 221, 263, 266]. Similar to all these prior methods, our approach will also use a modified reward function to train the policy, but it will also modify how the model is trained such that the model and policy optimize the same objective. A third strategy is to directly optimize the model such that it produces good policies [6, 40, 170, 176, 224], as theoretically analyzed in Grimm et al. [82]. While our aim is the same as these prior methods, our approach will not require differentiating through unrolled model updates or optimization procedures.

Our work builds on prior work that proposes model-based RL objectives that are lower bounds on the true, expected returns. Kearns and Singh [114] provide a lower bound that holds globally, but is only computable in tabular settings. Luo et al. [150] provide a lower bound that can be efficiently estimated, but which only holds for nearby policies and models. Our lower bound combines the strengths of these prior works, providing a lower bound that holds globally and can

be efficiently estimated in MDPs with continuous states and actions. Unlike any lower bounds from prior work, ours mends the objective mismatch problem.

Our theoretical derivation builds on prior work that casts model-based RL as a two-player game between a model-player and a policy-player [15, 171, 192, 196]. However, whereas prior work pits model and policy compete against one another, our formulation will result in a cooperative game, wherein the model and policy players cooperate in optimizing the *same* objective (a lower bound on the expected return). Our approach, though structurally resembling a GAN, is different from prior work that simply replaces a maximum likelihood model with a GAN model [16, 31, 124].

The most similar prior work is VMBPO [37]. The mechanics of our method are similar, also learning a dynamics model using a classifier that distinguishes real versus generated rollouts. However, while our method maximizes a lower bound on expected return, VMBPO maximizes a different, risk-seeking objective, which is an *upper* bound on expected return. This different objective can be expressed as the expected return plus the variance of the return, so VMBPO has the undesirable property of preferring policies that receive slight lower return if the variance of the return is much larger (see Appendix d.1.1). Indeed, most of the components of our method, including classifiers and GAN-like models, have been used in prior work, main contribution of our work is a precise recipe for combining these components in a way that provably maximizes expected return.

5.3 A UNIFIED OBJECTIVE FOR MODEL-BASED RL

Notation. We focus on the Markov decision process with states s_t, actions a_t, initial state distribution $p_0(s_0)$, positive reward function $r(s_t, a_t) > 0$, and dynamics $p(s_{t+1} \mid s_t, a_t)$. Our aim is to learn a control policy $\pi_\theta(a_t \mid s_t)$ with parameters θ that maximizes the expected discounted return:

$$\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{5.1}$$

We use transitions (s_t, a_t, r_t, s_{t+1}) collected from the (real) environment to train the dynamics model $q_\theta(s_{t+1} \mid s_t, a_t)$, and use transitions sampled from this learned model to train the policy. To simplify notation, we will define a trajectory τ to be the sequence of states and actions visited in an episode: $\tau \triangleq (s_0, a_0, s_1, a_1, \cdots)$. We then define $R(\tau) \triangleq \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ as the discounted return of a trajectory. Finally, we define two distributions over trajectories. First, $p^\pi(\tau)$ is the distribution over trajectories when policy π_θ interacts with dynamics

$p(s' \mid s, a)$; $q^\pi(\tau)$ is the distribution over trajectories when policy π_θ interacts with the learned dynamics $q_\theta(s_{t+1} \mid s_t, a_t)$:

$$p^\pi(\tau) = p_0(s_0) \prod_{t=0}^{\infty} p(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t),$$
$$q^\pi(\tau) = p_0(s_0) \prod_{t=0}^{\infty} q_\theta(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t).$$

Desiderata. Our aim is to design an objective $\mathcal{L}(\theta)$ that can be used to *jointly* optimize both the policy ($\pi_\theta(a_t \mid s_t)$) and the dynamics model ($q(s_{t+1} \mid s_t, a_t)$), and which is a lower bound on the expected return in the true environment.

An objective for model-based RL. We now introduce an objective that achieves these aims. Our objective will be the policy's reward when interacting with the learned model, but using a different reward function. The new reward function augments the task reward with an additional term that measures the difference between the learned model and the real environment. We define our objective

$$\mathcal{L}(\theta) \triangleq \mathbb{E}_{q^{\pi_\theta}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(s_t, a_t, s_{t+1}) \right], \tag{5.2}$$

where the modified reward function is defined as

$$\tilde{r}(s_t, a_t, s_{t+1}) \triangleq (1-\gamma) \log r(s_t, a_t) + \log \left(\frac{p(s_{t+1} \mid s_t, a_t)}{q(s_{t+1} \mid s_t, a_t)} \right) - (1-\gamma) \log(1-\gamma). \tag{5.3}$$

This objective maximizes an *augmented* reward under the *learned* dynamics. The augmented reward function $\tilde{r}$ penalizes the policy for taking transitions that are unlikely under the true dynamics model, similar to prior work [52, 263]. Later, we will show that we can estimate this augmented reward *without knowing the true environment dynamics* by using a GAN-like classifier. We will optimize this lower bound with respect to both the policy $\pi_\theta(a_t \mid s_t)$ and the dynamics model $q_\theta(s_{t+1} \mid s_t, a_t)$. For the policy, this optimization entails performing RL to maximize the modified reward using samples from the learned model; the only difference from prior work is the modification to the reward function. Training the dynamics model using this objective is very different from standard maximum likelihood training, and instead resembles a GAN. The model is optimized to sample trajectories that both have high reward (i.e., $\log r$ is large) and are similar to real dynamics (i.e., $\log \frac{p}{q}$ is large). This objective differs from VMBPO [37] by taking the $\log(\cdot)$ of the original reward functions; our experiments demonstrate that excluding this component invalidates our lower bound and results in learning suboptimal policies.

Our objective has two properties that make it particularly useful. First, the model and the policy are trained using exactly the same objective: updating the model not only increases the objective for the model, but also increases the objective for the policy. Note that this is very different from prior work, where training the model to be more accurate (increase likelihood) can decrease the policy's expected return under that model. Second, our objective is a lower bound on the expected return. This property gives us a guarantee on how well the learned policy will perform when deployed on the real environment. To state this result formally, we will take the logarithm of the expected return. Of course, maximizing the $\log(\cdot)$ of the expected return is equivalent to maximizing the expected return.

Theorem 3. *The following bound holds for* ***any*** *dynamics* $q(s_{t+1} \mid s_t, a_t)$ *and policy* $\pi(a_t \mid s_t)$*:*

$$\log \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \geq \mathcal{L}(\theta).$$

The proof is presented in Appendix d.1.3. Note that the expected return under the learned model, which most prior model-based RL methods use to train the policy, is not a lower bound on the expected return. To the best of our knowledge, this is the first global (unlike Luo et al. [150]) and efficiently-computable (unlike Kearns and Singh [114]) lower bound for model-based RL.

Sec. 5.4 will introduce an algorithm to maximize this lower bound. While this lower bound may not be tight, experiments in Sec. 5.5 demonstrate that optimizing this first lower bound yields policies that achieve high reward across a wide range of tasks.

TIGHTENING THE LOWER BOUND. We now introduce a modification to our lower bound that does make the bound tight. This new lower bound will be more complex than the one introduced above and we have not yet successfully designed an algorithm for maximizing it. Nonetheless, we believe that presenting the bound may prove useful for the design of future model-based RL algorithms.

We will use $\mathcal{L}_\gamma(\theta)$ to denote this new lower bound. In addition to the policy and dynamics, this bound will also depend on a time-varying discount, $\gamma_\theta(t)$, in place of the typical γ^t term. Similar learned discount factors have been studied in previous work on model-free RL [198]. We define this objective as follows:

$$\mathcal{L}_\gamma(\theta) \triangleq \mathbb{E}_{q^{\pi_\theta}(\tau)} \left[\sum_{t=0}^{\infty} \gamma_\theta(t) \tilde{r}_\gamma(s_t, a_t, s_{t+1}) \right], \tag{5.4}$$

where the augmented reward is now defined as

$$\tilde{r}_\gamma(s_t, a_t, s_{t+1}) \triangleq \log r(s_t, a_t) + \frac{1 - \Gamma_\theta(t-1)}{\gamma_\theta(t)} \log \left(\frac{p(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \cdots)}{q_\theta(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \cdots)} \right) + \log \left(\frac{\gamma^t}{\gamma_\theta(t)} \right),$$

and $\Gamma_\theta(t) = \sum_{t'=0}^{t} \gamma_\theta(t')$ is the CDF of the learned discount function (i.e., $\gamma_\theta(t)$ is a probability distribution over t.). This new lower bound, which differs from our main lower bound by the learnable discount factor, does provide a tight bound on the expected return objective:

Lemma 4. *Let an arbitrary policy $\pi(a_t \mid s_t)$ be given. The objective $\mathcal{L}_\gamma(\theta)$ is also a lower bound on the expected return objective, $\log \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \geq \mathcal{L}_\gamma(\theta)$, and this bound becomes tight at optimality:*

$$\log \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] = \max_{q^\pi(\tau), \gamma_\theta(t)} \mathcal{L}_\gamma(\theta).$$

The proof is presented in Appendix d.1.4. One important limitation of this result is that the learned dynamics that maximize this lower bound to make the bound tight may be non-Markovian. Intriguingly, this analysis suggests that using non-Markovian models, such as RNNs and transformers, may accelerate learning on Markovian tasks. This work does not propose an algorithm for optimizing this more complex lower bound.

THE OPTIMAL DYNAMICS ARE OPTIMISTIC. We now return to analyzing the simpler lower bound ($\mathcal{L}(\theta)$ in Eq. 5.2). In stochastic environments, the dynamics model that optimizes this lower bound is not equal to the true environment dynamics. Rather, it is biased towards sampling trajectories with high return. Ignoring parametrization constraints, the dynamics model that optimizes our lower bound is $q^*(\tau) = \frac{p(\tau)R(\tau)}{\int p(\tau')R(\tau')d\tau'}$ (proof in Appendix d.1.4.). We hypothesize that the optimism in the dynamics model will accelerate policy optimization, a hypothesis we test in Sec. 5.5.1.

Would the optimistic dynamics overestimate the policy's return, violating Lemma 3? This is not quite what our method does. Rather, our method estimates the *augmented* reward using the optimistic dynamics model, and the reward augmentation compensates for the optimism in the dynamics model.

5.4 MISMATCHED NO MORE

The previous section presented a single (global) lower bound ($\mathcal{L}$ from Eq. 5.2) for jointly optimizing the policy and the dynamics model. In this section, we develop a practical algorithm for optimizing this

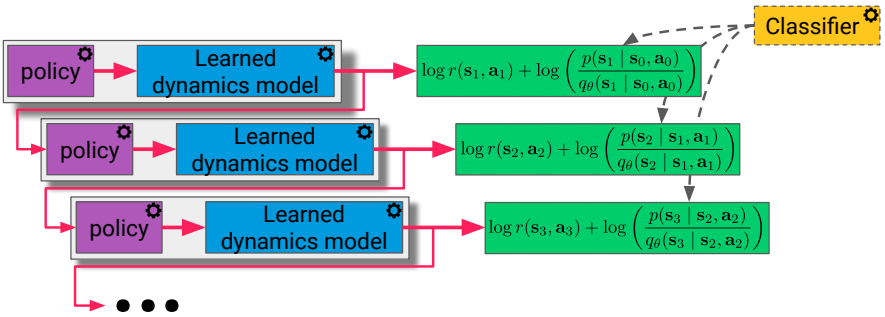


Figure 5.1: **Mismatched No More** is a model-based RL algorithm that learns a policy, dynamics model, and classifier. The classifier distinguishes real transitions from model transitions. The policy and dynamics model are jointly optimized to sample transitions that yield high return and look realistic, as estimated by the classifier.

lower bound. We call our method MISMATCHED NO MORE (MnM) because the policy and model optimize the same objective, thereby resolving the objective mismatch problem noted in prior work. The main challenge in optimizing this bound is that the augmented reward function depends on the transition probabilities of the real environment, $p(s_{t+1} \mid s_t, a_t)$, which are unknown. We address this challenge by learning a classifier (Sec. 5.4.1), and then describe the precise update rules for the policy, dynamics model, and classifier (Sec. 5.4.2).

5.4.1 *Estimating the Augmented Reward Function*

To estimate the augmented reward function, which depends on the transition probabilities of the real environment, we learn a classifier that distinguishes real transitions from fake transitions. This approach is similar to GANs [79] and similar to prior work in RL [52, 263]. We use $C_\phi(s_t, a_t, s_{t+1}) \in [0, 1]$ to denote the classifier. For an optimal classifier, we can use the classifier's predictions to estimate the augmented reward function:

$$\tilde{r}(s_t, a_t, s_{t+1}) = \log r(s_t, a_t) + \underbrace{\log \frac{C_\phi(s_t, a_t, s_{t+1})}{1 - C_\phi(s_t, a_t, s_{t+1})}}_{\approx \log p(s_{t+1}|s_t,a_t) - \log q_\theta(s_{t+1}|s_t,a_t)} . \tag{5.5}$$

The approximation above reflects function approximation error in learning the classifier.

We now present our complete method, which trains three components: a classifier, a policy, and a dynamics model. Our method alternates between *(1)* updating the policy (by performing RL using model experience with augmented rewards) and *(2)* updating the dynamics model and classifier (using a GAN-like objective). In de-

scribing the loss functions below, we use the superscripts $(\cdot)^{\text{real}}$ and $(\cdot)^{\text{model}}$ to denote transitions that have been sampled from the true environment dynamics or the learned dynamics function. To reduce clutter, we omit the superscripts when unambiguous.

5.4.2 *Updating the Model, Policy, and Classifier*

UPDATING THE POLICY. The policy is optimized to maximize the augmented reward on transitions sampled from the learned dynamics model. While this optimization can be done using any RL algorithm, including on-policy methods, we will focus on an off-policy actor-critic method.

We define the Q function as sum of *augmented* rewards under the learned dynamics model:

$$Q(s_t, a_t) \triangleq \mathbb{E}_{\substack{\pi(a_t|s_t),\\ q_\theta(s_{t+1}|s_t,a_t)}} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \tilde{r}(s_{t'}, a_{t'}) \mid s_t = s_t, a_t = a_t \right]. \quad (5.6)$$

We approximate the Q function using a neural network $Q_\psi(s_t, a_t)$ with parameters ϕ. We train the Q function using the TD loss on transitions sampled from the *learned* dynamics model:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}^{\text{model}}; \psi) = \left(Q_\psi(s_t, a_t) - \lfloor y_t \rfloor_{\text{sg}}\right)^2, \quad (5.7)$$

where $\lfloor \cdot \rfloor_{\text{sg}}$ is the stop-gradient operator and $y_t = \tilde{r}(s_t, a_t, s_{t+1}^{\text{model}}) + \gamma \mathbb{E}_{\pi(a_{t+1}|s_{t+1}^{\text{model}})} \left[Q_\psi(s_{t+1}^{\text{model}}, a_{t+1}) \right]$. The augmented reward, $\tilde{r}$, is estimated using the learned classifier. To estimate the corresponding value function, we use a 1-sample approximation: $V_\psi(s_t) = Q_\psi(s_t, a_t \sim \pi_\theta(a_t \mid s_t))$. The policy is trained to maximize the expected future (augmented) return, as estimated by the Q function:

$$\max_\theta \mathcal{L}_\pi(s_t; \theta) \triangleq \mathbb{E}_{\pi_\theta(a_t|s_t)} \left[\tilde{Q}_\psi(s_t, a_t) \right]. \quad (5.8)$$

In our implementation, we regularize the policy by adding an additional entropy regularizer. Following prior work [74], we maintain two Q functions and two target Q functions, use the minimum of the two target Q functions to compute the TD target.

UPDATING THE CLASSIFIER. We train the classifier to distinguish real versus model transitions using the standard cross entropy loss:

$$\max_\phi \mathcal{L}_C(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{real}}, s_{t+1}^{\text{model}}; \phi) \triangleq \log C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{real}}) + \log \left(1 - C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})\right). \quad (5.9)$$

Note that the real transition $(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{real}})$ and model transition $(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})$ have the same initial state and initial action.

Algorithm 6 Mismatched no More (MnM) is an algorithm for model-based RL. The method alternates between training the policy on experience from the learned dynamics model with augmented rewards and updating the model+classifier using a GAN-like loss. Updates are gradient steps with the Adam optimizer.

1: **while** not converged **do**
2: Sample experience from learned model and modify rewards using the classifier (Eq. 5.5).
3: Update policy and Q function using the model experience and augmented rewards (Eq.s 5.8 and 5.7).
4: Update model and classifier using GAN-like losses (Eq.s 5.9 and 5.10).
5: (Infrequently) Sample experience from real model.
6: **return** policy $\pi_\theta(a_t \mid s_t)$.

UPDATING THE DYNAMICS MODEL. To optimize the dynamics model, we rewrite the lower bound in terms of a single transition (derivation in Appendix d.1.6):

$$\mathcal{L}_q(s_t^{\text{real}}, a_t^{\text{real}}; \theta) = \mathbb{E}_{s_{t+1}^{\text{model}} \sim q_\theta(s_{t+1} \mid s_t^{\text{real}}, a_t^{\text{real}})} \left[V_\psi(s_{t+1}^{\text{model}}) + \log \frac{C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})}{1 - C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})} \right]. \tag{5.10}$$

The approximation above reflects approximation error in learning the optimal classifier. This approximation is standard in prior work on GANs [79] and adversarial inference [44, 48]. The procedure for optimizing the dynamics model and the classifier resembles a GAN [79]: the classifier is optimized to distinguish real transitions from model transitions, and the model is updated to fool the classifier (and increase rewards). However, *our method is not equivalent to simply replacing a maximum likelihood model with a GAN model*. Indeed, such an approach would not optimize a lower bound on expected return. Rather, our model objective includes an additional value term and our policy objective includes an additional classifier term. These changes enable the model and policy to optimize the same objective, which is a lower bound on expected return.

ALGORITHM SUMMARY. We summarize the method in Alg. 6 and provide an illustration in Fig. 5.1. Implementing MnM on top of a standard model-based RL algorithm is straightforward. First, create an additional classifier network. Second, instead of using the maximum likelihood objective to train the model, use the GAN-like objective in Eq. 5.10 to update both the model and the classifier. Third, add the classifier's logits to the predicted rewards (Eq. 5.5). Following prior

work [105], we learn a neural network to predict the true environment rewards.

5.5 EXPERIMENTS

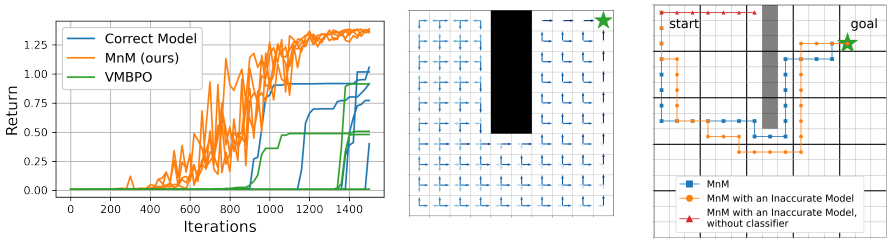
We present two sets of experiments. Our first set of experiments studies the importance of different components of MnM. Second, we study how MnM compares with prior model-based RL algorithms on challenging, robotic control tasks. To compare policies learned by different algorithms, we will evaluate the policies using the true environment dynamics, not the learned dynamics model.

5.5.1 *Understanding the Lower Bound and the Learned Dynamics*

In this section, we begin by studying the seemingly contradictory attributes of our method: optimistic dynamics and pessimistic policies, and end by confirming that together these components optimize an increasingly tight lower bound on the expected return.

Our theory suggests that MnM should work best in settings with stochastic dynamics and challenging exploration requirements, as the dynamics model should tilt the true dynamics to make the stochasticity more favorable for solving the task. We use a 10x10 gridworld with highly stochastic dynamics and a sparse reward function. The results, shown in Fig. 5.2a *(Left)*, show that MnM outperforms both Q-learning and VMBPO. In line with our theory, the dynamics learned by MnM (Fig. 5.2a *(Right)*) alter the environment stochasticity to lead the agent towards the goal, increasing the probability of collecting high-reward experience. Of course, we use the true environment dynamics, not the optimistic dynamics model, for evaluating the policies. While VMBPO also learns optimistic dynamics, it omits log-transformation of the reward function (which encourages pessimistic behavior), a difference that has a large effect on this task.

Our augmented reward function contains two crucial components, *(1)* the classifier term and *(2)* the logarithmic transformation of the reward function. We test the importance of the classifier term in correcting for inaccurate models. To do this, we limit the capacity of the MnM dynamics model so that it makes "low-resolution" predictions, forcing all states in 3×3 blocks to have the same dynamics. We will use the gridworld shown in Fig. 5.2b, which contains obstacles that occur at a finer resolution than the model can detect. When the "low resolution" dynamics model makes predictions for states near the obstacle, it will average together some states with obstacles and some states without obstacles. Thus, the model will (incorrectly) predict that the agent always has some probability of moving in each direction, even if that direction is actually blocked by an obstacle. However, the classifier (whose capacity we have not limited) detects that the



(a) Stochastic Gridworld (b) Inaccurate Models

Figure 5.2: **Two Didactic Experiments.** *(Left)* We apply MnM to a navigation task with transition noise that moves the agent to neighboring states with equal probability. MnM solves this task more quickly than Q-learning and VMBPO. The dynamics learned by MnM are different from the real dynamics, changing the transition noise (blue arrows) to point towards the goal. *(Right)* We simulate function approximation by a learning model that is forced to make the same predictions for groups of 3×3 states, resulting in a model that is inaccurate around obstacles. The classifier term compensates for this function approximation error by penalizing the policy for navigating near obstacles.

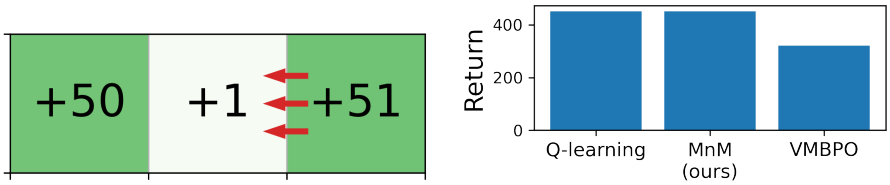


Figure 5.3: **Testing for risk seeking behavior**: On a simple 3-state MDP with stochastic transition in one state (red arrows), MnM converges to the reward-maximizing policy while VMBPO learns a strategy with lower rewards and higher variance (as predicted by theory).

dynamics model is inaccurate in these states and so the augmented reward is much lower at these states. Thus, MnM is able to solve this task despite the inaccurate model; an ablation of MnM that removes the classifier term attempts to navigate through the wall and fails to reach the goal.

We then test the importance of the logarithmic transformation by comparing MnM to VMBPO, which includes the classifier term but omits the logarithmic transformation. We hypothesize that VMBPO's deviation from our lower bound will cause it to exhibit risk seeking behavior. To test this hypothesis, we use the 3-state MDP in Fig. 5.3 *(top)* where numbers indicate the reward at each state. While moving to the right state yields slightly higher rewards, "wind" knocks the agent out of this state with probability 50% so the reward-maximizing strategy is to move to the left state. While MnM MnM learns the reward-maximizing strategy, VMBPO learns a policy that goes to the right state and receives lower returns (with much higher variance).

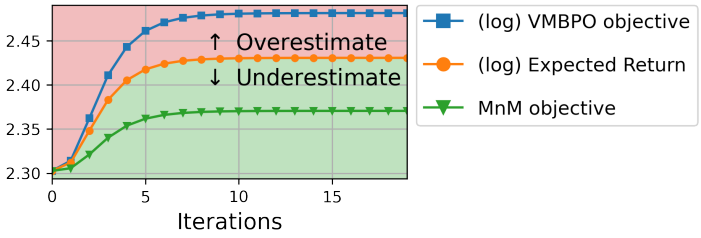


Figure 5.4: **Comparing objectives**: We apply value iteration to the gridworld from Fig. 5.2a to analytically compute various objectives. As predicted by our theory, the MnM objective is a lower bound on the expected return, whereas the VMBPO objective overestimates the expected return.

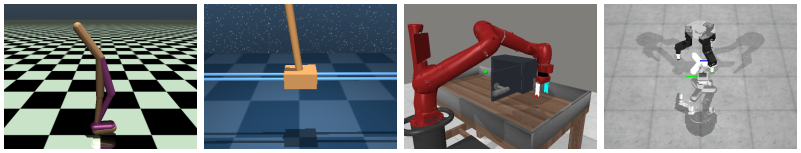


Figure 5.5: **Environments**: Our experiments included tasks from four benchmarks: (clockwise from top-left) OpenAI Gym, DM Control, Metaworld, and ROBEL.

Finally, we study how the MnM objective compares to alternative objectives. We use the gridworld from Fig. 5.2a and use a version of MnM based on value iteration to avoid approximation error. Plotting the MnM objective in Fig. 5.4, we observe that it is always a lower bound on the (log) expected return, as predicted by our theory. Ablations of MnM to omit the reward augmentation or even just the log transformation (i.e., VMBPO) overestimate the expected return.

5.5.2 *Comparisons On Robotics Tasks*

Our next experiments use continuous-control robotic tasks to answer two questions. We first investigate whether MnM performs at least comparably with prior work. We then study tasks with sparse rewards and more challenging exploration, where we suspect the optimistic dynamics learned by MnM may be beneficial. We illustrate a subset of the environments in Fig. 5.5. One detail of note is that we omit the reward augmentation (Eq. 5.3) for MnM during these experiments, as it hinders exploration leading to lower returns. We use MBPO [105] as a baseline for model-based RL because it achieves state-of-the-art results and is a prototypical example of model-based RL algorithms that use maximum likelihood models.

Our first comparison uses three locomotion tasks from the OpenAI Gym benchmark [26], which has become the standard benchmark for model-based RL algorithms. Thes tasks have dense rewards and pose no significant exploration challenge, so we do not expect MnM

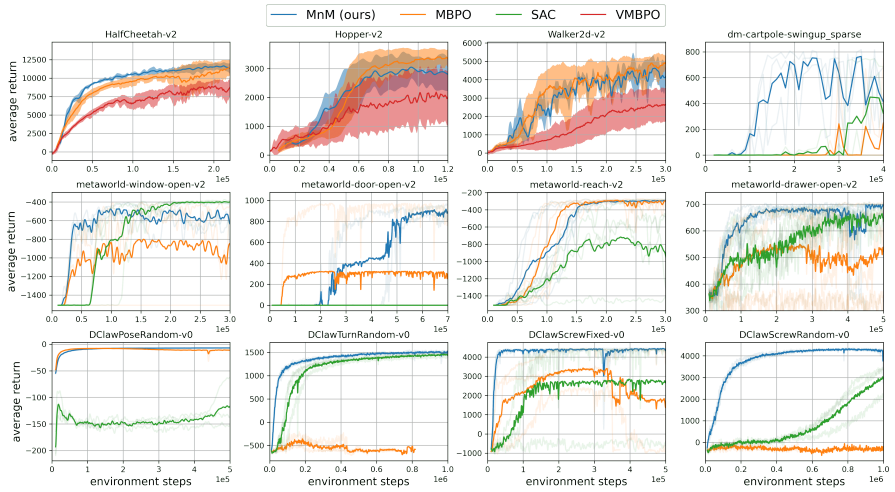


Figure 5.6: **Comparison on Robotics Tasks**: We compare MnM to MBPO and SAC on simulated control tasks. On the benchmark locomotion tasks *(top left)*, MnM performs comparably with MBPO. On many of the other tasks with sparse rewards that pose an exploration challenge, MnM outperforms both MBPO and the model-free baseline. These experiments suggest that maximizing a well defined bound on expected return, as done by our method, can lead to improved performance on difficult tasks. Transparent lines depict individual random seeds.

to outperform prior methods. The results (Fig. 5.6 *(top left)*) show that MnM performs roughly on par with MBPO. For all the plots in Fig. 5.6, the dark line indicates the average across random seeds, each of which is shown as a transparent line. For the tasks comparing to MBPO, we shade one standard deviation around the mean instead of showing separate learning curves, as these are not available for the MBPO baseline.

Tasks with sparse rewards, complex contact dynamics, and those requiring hard exploration often present a challenge for model-based RL algorithms, which are liable to exploit inaccuracies in the learned dynamics model. Our next experiment evaluates MnM on control tasks used in prior work that demonstrate these properties. These tasks the sparse-reward cartpole task from the DM Control benchmark [236], four manipulation tasks from the Metaworld benchmark [265], and four dexterous manipulation tasks from the ROBEL benchmark [4]. The results shown in Fig. 5.6 indicate that the complex environment dynamics of these tasks can cause prior model-based algorithms (MBPO) to perform worse than model-free algorithms (SAC), both in terms of asymptotic performance and sample complexity. Nonetheless, we observe that MnM frequently outperforms all prior methods and, more importantly, it *consistently* does well across all tasks. We include ablation experiments, including a comparison to VAML [61], in Appendix d.2.

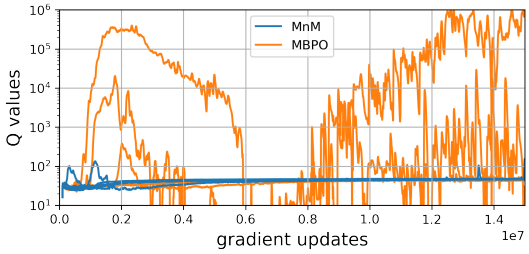


Figure 5.7: **Model exploitation**: The very large Q values of MBPO suggest model exploitation, which our method appears to avoid. Each line depicts a separate random seed.

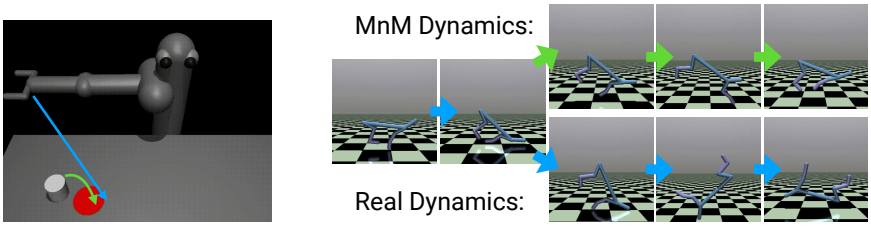


Figure 5.8: **Optimistic Dynamics**: *(Left)* On the `Pusher-v2` task, the MnM dynamics model makes the puck move towards the puck move towards the gripper before being grasped. *(Right)* On the `HalfCheetah-v2` task, the MnM dynamics model helps the agent stay upright after tripping.

To investigate the benefits of MnM over prior model-based methods, we logged the Q values throughout training and visualized them for the `metaworld-drawer-open-v2` task in Fig. 5.7. For fair comparison, we use Q values corresponding to just the task reward, omitting the logarithmic transformation and classifier term typically used by MnM. Fig. 5.7 shows that MnM yields Q values that are more accurate and more stable than MBPO. This figure suggests that MBPO may be exploiting inaccuracies in the learned model.

Finally, we visualize the dynamics learned by MnM on two robotic control tasks. These tasks have deterministic dynamics, so our theory would predict that an idealized version of MnM would learn a dynamics model exactly equal to the deterministic dynamics. However, our implementation relies on function approximation (neural networks) to learn the dynamics, and the limited capacity of function approximators makes otherwise-deterministic dynamics appear stochastic. On the `Pusher-v2` task, the MnM dynamics cause the puck to move towards the robot arm even before the arm has come in contact with the puck. While this movement is not physically realistic, it may make the exploration problem easier. On the `HalfChetah-v2` task, the MnM dynamics increase the probability that the agent remains upright after tripping, likely making it easier for the agent to learn how to run. We expect that the implicit stochasticity caused by function approximation to be especially important for real-world tasks, where the complexity

of the real dynamics often dwarfs the capacity of the learned dynamics model.

5.6 EXTENDING MNM TO LATENT-SPACE MODELS

In subsequent work [78], we extended MnM to better handle high-dimensional settings by developing a similar lower bound for latent-space models. The key to this construction was to treat both the observations and their representations as part of the latent variable, which is inferred by maximizing an evidence lower bound. Importantly, instead of inferring this entire latent variable, which would entail predicting high-dimensional observation, we used ideas similar to adversarial inference [44, 48] to optimize this bound without ever having to reconstruct the observations (i.e., our method does not require a decoder). Results showed that this method not only achieves good result, but does so at significantly lower computational costs than MnM and other model-based methods.

5.7 DISCUSSION

The main contribution of this work is an approach to model-based RL where the policy and dynamics model are jointly optimized using the same objective. Unlike prior work, our objective is a global lower bound on the standard expected return objective. Our approach not only tells users how to train their dynamics model, but also guarantees to them that updating their model (using our objective) will result in a better policy. We therefore believe that this *joint optimization* will ease and accelerate the design of future model-based RL algorithms. We suspect that the tools presented in this work may prove useful for solving tasks that require extensive exploration or long-horizon planning.

6 OUTLOOK

Taken together, the main message of this thesis is to think about decision making problems *not* in terms of scalar utility maximization, as suggested by the standard MDP formalism, but in terms of data and random variables. Outcomes can be specified via observations (Chapter 2) or sets of observations (Sec. 2.7). In essence, we use data to specify *what* at desired outcome looks like. This framing is practically useful, allowing a means for non-expert users to convey their intentions to ML systems (just snap a photo!). Moreover, it makes it easy to incorporate ideas and machinery from other areas of machine learning to accelerate and extend these methods. For example, Chapter 3 shows how contrastive learning ideas developed in other domains can readily be adapted into high-performance RL algorithms. More broadly, this framing allows us to use tools from probabilistic inference to reason about *how* to reach a desired outcome. What are the states and actions that lead from here to there? This was the question we explored in Part ii of the thesis.

As I write this, in the Spring of 2023, the machine learning community sits at a peculiar juncture: large language models have demonstrated such excellent results, yet require such high compute budgets, that many researchers are rethinking how to conduct research. How can academics build GPU clusters large enough to run these experiments? Do students need to be more highly trained in software engineering and computer systems to manage such experiments? Without large compute resources individually, should academics pool their resources into large, multi-institution collaborations? Should academics leave the large-scale training to industrial labs, and focus instead on different questions entirely (e.g., trust, dataset curation)?

Within the RL community, there are debates about whether language modeling techniques can be directly used to solve sequential decision making problems. Our preliminary analysis [59] suggests that it can (in theory), as long as these methods are normalized properly to handle hindsight bias. Regardless of their precise form, it seems likely that the next generation of RL algorithms are going to be ones that leverage massive datasets and massive compute. Likely, they will be built in a two-stage fashion, with compute-intensive training done once (in industry?) and lightweight, domain-specific adaptation performed for each downstream task (in academia?). The algorithms presented in Part i of this thesis are particularly amenable to this two-stage decomposition: contrastive representations are fit on large quantities of reward-free data, and then used to represent value functions using

one (or a handful) of observations from desired outcomes. If scaled appropriately, these might be called "contrastive foundation models." Unlike existing foundation models for CV and NLP, such contrastive foundation models might actually understand *time*. What effect does making a decision have? Why do certain objects only move when touched? Understanding time (and its close connection with causality) may not only yield better reinforcement learning methods, but also may provide representations that enable better supervised learning. Representations with a causal understanding of the world may be less likely to pick up on spurious features, and those representations trained with temporal difference learning may make better predictions about the distant future.

Scaling these methods, though, will likely require more than just increasing the dataset size and batch size. For one, sequences of observations (i.e., videos) are much cheaper to collect than action-labeled videos, so we likely will need to figure out how to extend these methods to make use of mixtures of action-free and action-labeled videos. Second, the recent large language models have demonstrated how large corpora of text contain vast amounts of information about the physical world; figuring out how to distill or embed such knowledge into contrastive foundation models remains an open question. Third, and finally, solving decision-making problems from data requires, well, data. Unlike language modeling, where even multi-lingual models only have to contend with tens of different languages, the vast number of possible decision making problems (from robotics to chemistry to medicine) means that solving many decision making problems will require active data collection. There has been much excellent work on active learning and intelligent exploration, but I think the key challenge here may be more societal than technical: collecting the scale of data needed for training decision-making systems tomorrow will likely require that these systems be widely deployed today. How can we construct decision-making tools that are actually useful (or at least not worse than their unlearned counterparts), so that we can start collecting the data for tomorrow's experiments? By making it easier for users to apply RL algorithms to real-world problems, I hope that this thesis takes a step towards this goal.

APPENDIX

a

LEARNING TO ACHIEVE GOALS VIA RECURSIVE CLASSIFICATION

A.1 A CONNECTION BETWEEN MAXIMIZING PROBABILITIES AND MINIMIZING DISTANCES

The overall objective of C-learning is to maximize the likelihood of the goal state under the discounted state occupancy measure; it is about maximizing a probability (density). This stands in contrast to prior work on stochastic shortest path problems, where the aim is to minimize the expected distance to the goal. Our motivation for using probabilities, rather than distances, is that distances can be ill-defined in settings where there is some probability of never reaching the goal. However, in settings where the goal is reached with probability one, we can directed relate these two objectives; this section explains this connection.

Assume an infinite horizon, tabular MDP. Assume that the commanded commanded goal state is an absorbing state; once the agent reaches the goal, it remains there indefinitely. Finally, assume that the expected time to reach the goal is finite for every policy. One way to ensure this condition is to assume that the MDP is ergodic, and to restrict the class of policies to those that add a bit of noise to their actions. Define Δ as a random variable denoting the expected number of steps to reach the goal; by construction, $\mathbb{E}[\Delta]$ is finite.

We can now use Δ to express the maximum probability objective:

$$\begin{aligned}
\max_\pi \mathcal{L}_{\text{prob}}(\pi) &= p^{\pi(\cdot|\cdot,g}(s_{t+} = g \mid s, a) \qquad \text{(maximum probability)} \\
&= (1-\gamma)\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \delta(s_t = g) \right] \\
&= (1-\gamma)\mathbb{E}_\Delta \left[\sum_{t=\Delta}^{\infty} \gamma^t \right] \\
&= \cancel{(1-\gamma)}\mathbb{E}_\Delta \left[\frac{\gamma^\Delta}{\cancel{1-\gamma}} \right] = \mathbb{E}_\Delta \left[\gamma^\Delta \right].
\end{aligned}$$

The shortest path objective can be expressed simply as

$$\max_\pi \mathcal{L}_{\text{dist}}(\pi) = \mathbb{E}_\Delta[-\Delta]. \qquad \text{(shortest path)}$$

We can use Jensen's inequality to relate (the logarithm of) the maximum probability objective to the shortest path objective:

$$\log \mathcal{L}_{\text{prob}}(\pi) \geq \log(1/\gamma) \cdot \mathbb{E}_{\Delta}\left[-\Delta\right] = \log(1/\gamma) \cdot \mathcal{L}_{\text{dist}}(\pi).$$

This inequality becomes tight when the policy always takes the same number of steps to reach the goal (e.g., deterministic policy with deterministic dynamics); when this happens, finding the shortest path is equivalent to the maximum probability objective.

In more general settings, this inequality means that the shortest path objective is a lower bound on maximum probability objective. Intuitively, this makes sense: by incorporating the logarithm, the shortest path objective effectively corresponds to a risk-sensitive objective [155]. For example, consider choosing between policy A (50% chance of reaching the goal in 2 steps, 50% chance of taking 100 steps to reach the goal) and policy B (always reaches the goal in 10 steps). For a reasonable discount of $\gamma = 0.9$, the maximum probability objective would prefer strategy A[1], whereas the shortest path objective would prefer strategy B.[2]

Should users prefer the maximum probability objective or the shortest path objective? The maximum probability objective is practically appealing because it avoids the need to manually define when the goal is reached, suggests algorithms where an important hyperparameter (the relabeling ratio) is determined by the theory, and suggests new goal-conditioned algorithms based on contrastive learning. The maximum probability objective is also theoretically appealing because it remains well defined in continuous settings, and in settings where the goal may never be reached. However, the analysis above shows that the maximum probability objective can be seen as a risk-seeking version of the the shortest path objective, assigning considerably lower weight to outcomes where the goal is never reached.

A.2 A BELLMAN EQUATION FOR C-LEARNING AND CONVERGENCE GUARANTEES

The aim of this section is to show that off-policy C-learning converges, and that the fixed point corresponds to the Bayes-optimal classifier. This result guarantees that C-learning will accurate *evaluate* the future state density of a given policy. We then provide a policy improvement theorem, which guarantees that goal-conditioned C-learning converges to the optimal goal-conditioned policy.

1 $\mathcal{L}_{\text{prob}}(A) = (1-0.9)(0.9^2 + 0.9^{100}) = 0.081; \mathcal{L}_{\text{prob}}(B) = (1-0.9)0.9^{10} = 0.035.$

2 $\mathcal{L}_{\text{dist}}(A) = -\frac{1}{2}(2+100) = -51; \mathcal{L}_{\text{dist}}(B) = -10.$

A.2.1 *Bellman Equations for C-Learning*

We start by introducing a new Bellman equation for C-learning, which will be satisfied by the Bayes optimal classifier. While actually evaluating this Bellman equation requires privileged knowledge of the transition dynamics and the marginal state density, if we knew these quantities we could turn this Bellman equation into a convergent value iteration procedure. In the next section, we will show that the updates of off-policy C-learning are equivalent to this value iteration procedure, but do not require knowledge of the transition dynamics or marginal state density. This equivalence allows us to conclude that C-learning converges to the Bayes-optimal classifier.

Our Bellman equation says that the future state density function f_θ induced by a classifier C_θ should satisfy the recursive relationship noted in Eq. 2.4.

Lemma 5 (C-learning Bellman Equation)**.** *Let policy $\pi(\mathbf{a_t} \mid \mathbf{s_t})$, dynamics function $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$, and marginal distribution $p(\mathbf{s_{t+}})$ be given. If a classifier C_θ is the Bayes-optimal classifier, then it satisfies the follow identity for all states $\mathbf{s_t}$, actions $\mathbf{a_t}$, and potential future states $\mathbf{s_{t+}}$:*

$$\frac{C^\pi_\theta(F=1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})} = (1-\gamma)\frac{p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})}{p(\mathbf{s_{t+}})} + \gamma \mathbb{E}_{\substack{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}),\\ \pi(\mathbf{a_{t+1}}|\mathbf{s_t})}}\left[\frac{C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}\right] \tag{a.1}$$

Proof. If C_θ is the Bayes-optimal classifier, then $f^\pi_\theta(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) = p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$. Substituting the definition of f_θ (Eq. 2.2) into Eq. 2.4, we obtain a new Bellman equation: □

This Bellman equation is similar to the standard Bellman equation with a goal-conditioned reward function $r_{\mathbf{s_{t+}}}(\mathbf{s_t}, \mathbf{a_t}) = p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})/p(\mathbf{s_{t+}})$, where Q functions represent the ratio $f^\pi_\theta(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) = p^\pi_+(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$. However, actually computing this reward function to evaluate this Bellman equation requires knowledge of the densities $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$ and $p(\mathbf{s_{t+}})$, both of which we assume are unknown to our agent.[3] Nonetheless, if we had this privileged information, we could readily turn this Bellman equation into the following assignment equation:

$$\begin{aligned}\frac{C^\pi_\theta(F=1 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+}})} \leftarrow (1-\gamma)&\frac{p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})}{p(\mathbf{s_{t+}})} \\ &+ \gamma \mathbb{E}_{\substack{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}),\\ \pi(\mathbf{a_{t+1}}|\mathbf{s_t})}}\left[\frac{C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}, \mathbf{s_{t+}})}\right]\end{aligned} \tag{a.2}$$

3 Interestingly, we can efficiently estimate this reward function by learning a *next-state* classifier, $q_\theta(F \mid \mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})$, which distinguishes $\mathbf{s_{t+1}} \sim p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$ from $\mathbf{s_{t+1}} \sim p(\mathbf{s_{t+1}}) = p(\mathbf{s_{t+}})$. This classifier is *different* from the future state classifier used in C-learning. The reward function can then be estimated as $r_{\mathbf{s_{t+}}}(\mathbf{s_t}, \mathbf{a_t}) = \frac{q_\theta(F=1|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{q_\theta(F=0|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}$. If we learned this next state classifier, we estimate the future state density and learn goal-reaching policies by applying standard Q-learning to this reward function.

Lemma 6. *If we use a tabular representation for the* ratio $\frac{C^\pi_\theta(F=1|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{C^\pi_\theta(F=0|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}$*, then iterating the assignment equation (Eq. a.2) converges to the optimal classifier.*

Proof. Eq. a.2 can be viewed as doing value iteration with a goal-conditioned Q function parametrized as $Q(\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}) = \frac{C^\pi_\theta(F=1|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{C^\pi_\theta(F=0|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}$ and a goal-conditioned reward function $r_{\mathbf{s_{t+}}}(\mathbf{s_t},\mathbf{a_t}) = \frac{p(\mathbf{s_{t+1}}=\mathbf{s_{t+}}|\mathbf{s_t},\mathbf{a_t})}{p(\mathbf{s_{t+}})}$. We can then employ standard convergence proofs for Q-learning to guarantee convergence [104, Theorem 1]. □

A.2.2 *Off-Policy C-learning Converges*

In this section we show that off-policy C-learning converges to the Bayes-optimal classifier, and thus recovers the true future state density function. The main idea is to show that the updates for off-policy C-learning have the same effect as the assignment equation above (Eq. a.2), without relying on knowledge of the dynamics function or marginal density function.

Lemma 7. *Off-policy C-learning results in the same updates to the classifier as the assignment equations for the C-learning Bellman equation (Eq. a.2)*

Proof. We start by viewing the off-policy C-learning loss (Eq. 2.10) as a *probabilistic* assignment equation. A given triplet $(\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})$ can appear in Eq. 2.10 in two ways:

1. We sample a "positive" $\mathbf{s_{t+}} = \mathbf{s_{t+1}}$, which happens with probability $(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t})$, and results in the label $y = 1$.
2. We sample a "negative" $\mathbf{s_{t+}}$, which happens with probability $(1+\gamma w)p(\mathbf{s_{t+}})$ and results in the label $y = \frac{\gamma w}{\gamma w+1}$.

Thus, conditioned on the given triplet containing $\mathbf{s_{t+}}$, the expected target value y is

$$\begin{aligned}\mathbb{E}[y \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}] &= \frac{(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t})\cdot 1 + \mathbb{E}\left[\cancel{(1+\gamma w)}p(\mathbf{s_{t+}})\cdot\cancel{\frac{\gamma w}{\gamma w+1}}\right]}{(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t}) + \mathbb{E}\left[(1+\gamma w)p(\mathbf{s_{t+}})\right]} \\ &= \frac{(1-\gamma)\frac{p(\mathbf{s_{t+1}}=\mathbf{s_{t+}}|\mathbf{s_t},\mathbf{a_t})}{p(\mathbf{s_{t+}})} + \gamma\mathbb{E}[w]}{(1-\gamma)\frac{p(\mathbf{s_{t+1}}=\mathbf{s_{t+}}|\mathbf{s_t},\mathbf{a_t})}{p(\mathbf{s_{t+}})} + \gamma\mathbb{E}[w] + 1}. \end{aligned} \tag{a.3}$$

Note that w is a random variable because it depends on $\mathbf{s_{t+1}}$ and $\mathbf{a_{t+1}}$, so we take its expectation above. We can write the assignment equation for C as

$$C^\pi_\theta(F = 1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}) \leftarrow \mathbb{E}[y \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}].$$

Noting that the function $\frac{C}{1-C}$ is strictly monotone increasing, the assignment equation is equivalent to the following assignment for the *ratio* $\frac{C^\pi_\theta(F=1|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{C^\pi_\theta(F=0|\mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}$:

$$\frac{C^\pi_\theta(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})} \leftarrow \frac{\mathbb{E}[y \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}]}{1-\mathbb{E}[y \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}]} = (1-\gamma)\frac{p(\mathbf{s_{t+1}}=\mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t})}{p(\mathbf{s_{t+}})} + \gamma\mathbb{E}[w].$$

The equality follows from substituting Eq. a.3 and then simplifying. Substituting our definition of w, we observe that the assignment equation for off-policy C-learning is exactly the same as the assignment equation for the C-learning Bellman equation (Eq. a.1):

$$\begin{aligned}\frac{C^\pi_\theta(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})} \leftarrow &(1-\gamma)\frac{p(\mathbf{s_{t+1}}=\mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t})}{p(\mathbf{s_{t+}})} \\ &+ \gamma\left[\frac{C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}},\mathbf{a_{t+1}},\mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_{t+1}},\mathbf{a_{t+1}},\mathbf{s_{t+}})}\right].\end{aligned}$$

□

Since the off-policy C-learning assignments are equivalent to the assignments of the C-learning Bellman equation, any convergence guarantee that applies to the later must apply to the former. Thus, Lemma 6 tells us that off-policy C-learning must also converge to the Bayes-optimal classifier. We state this final result formally:

Corollary 7.1. *If we use a tabular representation for the classifier, then off-policy C-learning converges to the Bayes-optimal classifier. In this case, the predicted future state density (Eq. 2.2) also converges to the true future state density.*

A.2.3 *Goal-Conditioned C-Learning Converges*

In this section we prove that the version of policy improvement done by C-learning is guaranteed to improve performance. We start by noting a Bellman *optimality* equation for goal-conditioned C-learning, which indicates whether a goal-conditioned policy is optimal:

Lemma 8 (C-learning Bellman *Optimality* Equation)**.** *Let dynamics function $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, ca)$, and marginal distribution $p(\mathbf{s_{t+}})$ be given. If a classifier C_θ is the Bayes-optimal classifier, then it satisfies the follow identity for all states $\mathbf{s_t}$, actions $\mathbf{a_t}$, and goals $g = \mathbf{s_{t+}}$:*

$$\begin{aligned}\frac{C^\pi_\theta(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})} =&(1-\gamma)\frac{p(\mathbf{s_{t+1}}=\mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t})}{p(\mathbf{s_{t+}})} \\ &+ \gamma\mathbb{E}_{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t})}\left[\max_{\mathbf{a_{t+1}}}\frac{C^\pi_\theta(F=1 \mid \mathbf{s_{t+1}},\mathbf{a_{t+1}},\mathbf{s_{t+}})}{C^\pi_\theta(F=0 \mid \mathbf{s_{t+1}},\mathbf{a_{t+1}},\mathbf{s_{t+}})}\right].\end{aligned} \tag{a.4}$$

We now apply the standard policy improvement theorem to C-learning.

Lemma 9. *If the estimate of the future state density is accurate, then updating the policy according to Eq. 2.9 guarantees improvement at each step.*

Proof. We use π to denote the current policy and π' to denote the policy that acts greedily w.r.t. the current density function:

$$\pi'(\mathbf{a_t} \mid \mathbf{s_t}, \mathbf{s_{t+}}) = \mathbb{1}(\mathbf{a_t} = \arg\max_a p^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}))$$

The proof is similar to the standard policy improvement proof for Q-learning.

$$\begin{aligned}
p^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}) &= \mathbb{E}_{\pi(\mathbf{a_t}|\mathbf{s_t},\mathbf{s_{t+}})}[p^\pi(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})] \\
&= \mathbb{E}_{\pi(\mathbf{a_t}|\mathbf{s_t},\mathbf{s_{t+}})}[(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) + \gamma p^\pi(\mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}})] \\
&\le \mathbb{E}_{\mathbf{g'}(\mathbf{a_t}|\mathbf{s_t},\mathbf{s_{t+}})}[(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) + \gamma p^\pi(\mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}})] \\
&= \mathbb{E}_{\pi'(\mathbf{a_t}|\mathbf{s_t},\mathbf{s_{t+}})}[(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) \\
&\quad + \mathbb{E}_{\substack{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}), \\ \pi(\mathbf{a_{t+1}}|\mathbf{s_{t+1}},\mathbf{s_{t+}})}} [\gamma((1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}) + \gamma p^\pi(\mathbf{s_{t+}} \mid s_{t+2}, a_{t+2}))]] \\
&\le \mathbb{E}_{\pi'(\mathbf{a_t}|\mathbf{s_t},\mathbf{s_{t+}})}[(1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t}) \\
&\quad + \mathbb{E}_{\substack{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}), \\ \mathbf{g'}(\mathbf{a_{t+1}}|\mathbf{s_{t+1}},\mathbf{s_{t+}})}} [\gamma((1-\gamma)p(\mathbf{s_{t+1}} = \mathbf{s_{t+}} \mid \mathbf{s_{t+1}}, \mathbf{a_{t+1}}) + \gamma p^\pi(\mathbf{s_{t+}} \mid s_{t+2}, a_{t+2}))]] \\
&\cdots \\
&\le p^{\pi'}(\mathbf{s_{t+}} \mid \mathbf{s_t})
\end{aligned}$$

□

Taken together with the convergence of off-policy C-learning, this proof guarantees that goal-conditioned C-learning converges to the optimal goal-reaching policy in the tabular setting.

A.3 MIXING TD C-LEARNING WITH MC C-LEARNING

Recall that the main challenge in constructing an off-policy procedure for learning the classifier was getting samples from the future state distribution of a *new* policy. Recall that TD C-learning (Alg. 3) uses importance weighting to estimate expectations under this new distribution, where the importance weights are computed using the learned classifier. However, this approach can result in high-variance, especially when the new policy has a future state distribution that is very different from the background distribution. In this section we describe how to decrease the variance of this importance weighting estimator at the cost of increasing bias.

The main idea is to combine TD C-learning with MC C-learning. We will modify off-policy C-learning to also use samples $\hat{p}(\mathbf{s_{t+}} \mid \mathbf{s_t}, \mathbf{a_t})$ from *previous policies* as positive examples. These samples will be sampled from trajectories in the replay buffer, in the same way that samples were generated for MC C-learning. We will use a mix of these on-policy samples (which are biased because they come from a different policy) and importance-weighted samples (which may have

higher variance). Weighting the TD C-learning estimator by λ and the MC C-learning estimator by $(1-\lambda)$, we get the following objective:

$$\begin{aligned}\lambda \mathbb{E}_{p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t}),p(\mathbf{s_{t+}})}[&(1-\gamma)\log C(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+1}}) + \log C(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}) \\ &+ \gamma w \log C(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}) \\ + (1-\lambda)\mathbb{E}_{\hat{p}(\widehat{\mathbf{s_{t+}}}|\mathbf{s_t},\mathbf{a_t}),p(\mathbf{s_{t+}})}[&\log C(F=1 \mid \mathbf{s_t},\mathbf{a_t},\widehat{\mathbf{s_{t+}}}) + \log C(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})]\end{aligned}$$

This method is surprisingly easy to implement. Given a batch of B transitions $(\mathbf{s_t},\mathbf{a_t})$, we label $\frac{\lambda}{2}B$ with the next state $\mathbf{s_{t+1}}$, $\frac{1}{2}B$ with a random state $\mathbf{s_{t+}} \sim p(\mathbf{s_{t+}})$, and $\frac{1-\lambda}{2}B$ with a state sampled from the empirical future state distribution $\hat{p}(\mathbf{s_{t+}} \mid \mathbf{s_t},\mathbf{a_t})$. To make sure that each term in the loss above receives the correct weight, we scale each of the terms by the inverse sampling probability:

$$\begin{aligned}\text{(Next states):} \quad & \frac{2}{\lambda B}(\lambda(1-\gamma)\log C(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+1}}) \\ \text{(Random states):} \quad & \frac{2}{B}\left((\lambda+1-\lambda)\log C(F=0 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}}) + \lambda\gamma w \log C(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})\right) \\ \text{(Future states):} \quad & \frac{2}{\cancel{(1-\lambda)}B}\cancel{(1-\lambda)}\log C(F=1 \mid \mathbf{s_t},\mathbf{a_t},\mathbf{s_{t+}})\end{aligned}$$

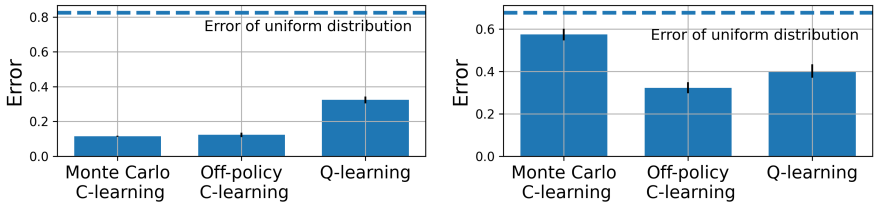
Without loss of generality, we scale each term by $\frac{B}{2}$. Since each of these terms is a cross entropy loss, we can simply implement this loss as a weighted cross entropy loss, where the weights and labels are given in the table below.

	Fraction of batch	Label	Weight
Next states	$\frac{\lambda}{2}$	1	$1-\gamma$
Future states	$\frac{1-\lambda}{2}$	1	1
Random states	$\frac{1}{2}$	$\frac{\lambda\gamma w}{1+\lambda\gamma w}$	$(1+\lambda\gamma w)$

On many tasks, we observed that this approach performed no differently than TD C-learning. However, we found this strategy to be crucial for learning some of the sawyer manipulation tasks. In our experiments we used $\lambda = 0.6$ for the Sawyer Push and Sawyer Drawer tasks, and used $\lambda = 1$ (i.e., pure TD C-learning) for all other tasks.

A.4 ADDITIONAL EXPERIMENTS

COMPARING C-LEARNING AND C-LEARNING FOR FUTURE STATE DENSITY ESTIMATION Fig. A.1 shows the results of our comparison of C-learning and Q-learning on the "continuous gridworld" environment, in both the on-policy and off-policy setting. In both settings, off-policy C-learning achieves lower error than Q-learning. As expected, Monte Carlo C-learning performs well in the on-policy



(a) On Policy (b) Off Policy

Figure a.1: We use C-learning and Q-learning to predict the future state distribution. *(Right)* In the on-policy setting, both the Monte Carlo and TD versions of C-learning achieve significantly lower error than Q-learning. *(Right)* In the off-policy setting, the TD version of C-learning achieves lower error than Q-learning, while Monte Carlo C-learning performs poorly, as expected.

setting, but poorly in the off-policy setting, motivating the use of off-policy C-learning.

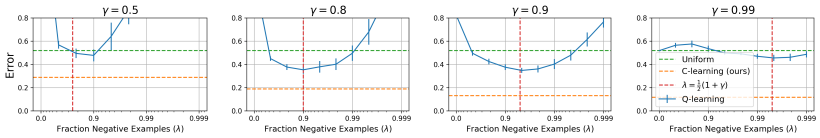


Figure a.2: The performance of Q-learning (blue line) is sensitive to the relabeling ratio. Our analysis accurately predicts that the optimal relabeling ratio is approximately $\lambda = \frac{1}{2}(1+\gamma)$. Our method, C-learning, does not require tuning this ratio, and outperforms Q-learning, even with the relabeling ratio for Q-learning is optimally chosen.

ADDITIONAL RESULTS ON PREDICTING THE GOAL SAMPLING RATIO To further test Hypothesis 2, we repeated the experiment from Fig. 2.2 across a range of values for γ. As shown in Fig. a.2, our Hypothesis accurately predicts the optimal goal sampling ratio across a wide range of values for γ.

A.5 PREDICTIONS FROM C-LEARNING

Figures a.3 and a.4 visualizes additional predictions from the C-learning model in Sec. 2.6. In each image, the top half shows the current state and the bottom half shows the predicted expected future state. Animations of these results can be found on the project website.

(a) HalfCheetah-v2, $\gamma = 0.9$

(b) HalfCheetah-v2, $\gamma = 0.9$

(c) Ant-v2, $\gamma = 0.5$

(d) Ant-v2, $\gamma = 0.9$

(e) Ant-v2, $\gamma = 0.99$

Figure a.3: Predictions from C-learning

(a) Walker2d-v2, $\gamma = 0.5$

(b) Walker2d-v2, $\gamma = 0.9$

(c) Walker2d-v2, $\gamma = 0.99$

(d) Hopper-v2, $\gamma = 0.5$

(e) Hopper-v2, $\gamma = 0.9$

Figure a.4: More Predictions from C-learning

b

CONTRASTIVE LEARNING AS GOAL-CONDITIONED REINFORCEMENT LEARNING

B.1 ADDITIONAL RELATED WORK

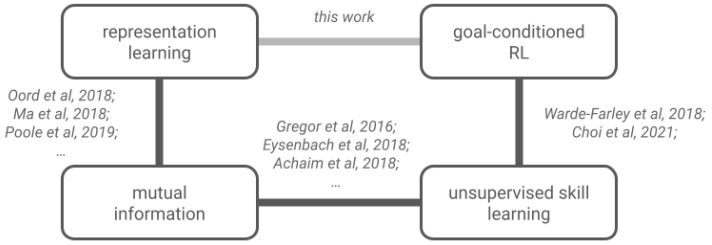


Figure b.1: **Connecting related work.** This work helps draw connections between prior work, filling in a missing link.

Our work is also related to unsupervised skill discovery [1, 35, 54, 80, 94, 130, 213], 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 [35, 248]. 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 [35], 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.

Fig. b.1 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 [153, 187]. Prior work in RL has proposed unsupervised skill learning algorithms using similar mutual information objectives [1, 54, 80], and more recent work has connected these unsupervised skills learning algorithms to goal-reaching. The key contribution of this effort is to connect representation learning to goal-conditioned RL.

B.2 DISCUSSION OF THE REPRESENTATIONS AS A MODEL

In contrastive RL, the critic predicts two representations $\phi(s)$, $\psi(g)$ whose inner product corresponds to the (goal-conditioned) value function. One way of thinking about these representations is that they form a sort of implicit model.[1]

One potential concern with these sorts of *simple* latent-space models is that they might fail to capture highly non-linear dynamics. In theory this is no object: if a non-linear latent-space model would perform better, we can construct an equivalent critic that does use a linear latent-space model (together with more expressive representations). One way of viewing this is that it pushes all of the representational burden onto the representations – it effectively turns the entire planning problem into a representation learning problem. From a practical perspective, our approach is appealing because representations are often pre-trained: large compute budgets can be spent on pre-training powerful representations on very large datasets, such that the resulting representations can be used in a compute-efficient manner (only inner products required).

Seen as a latent-space model, another potential concern with this form of a critic is that it does not explicitly capture the stochasticity of the dynamics – it only models the *expected* future representation, not the full distribution over future outcomes. One argument in favor of this is that only the expectation (not the full distribution) is required for selecting actions. However, it seems plausible that a proper distributional estimate might be easier to learn, or might be learned with higher fidelity (reminiscent of the success of distributional value functions for standard reward-maximization problems [19]). We leave this investigation to future work.

B.3 PROOFS

B.3.1 *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. 3.4):

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

1 This can be made precise by parameterizing $\phi(s) = A\psi(s) + b$; the matrix A and vector b now exactly correspond to a (linear) model of the representations.

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. b.1 as

$$p(s_{t+} = s_g) = (1-\gamma)p_0(s_0 = s_g) + (1-\gamma)\sum_{t=1}^{\infty}\gamma^t p_t^{\pi(\cdot|\cdot,s_g)}(s_t = s_g). \tag{b.2}$$

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. b.2, 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.3.2 *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 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 [59, Appendix C.2], where ϵ is the parameter for filtered relabeling (Sec. 3.4.5):

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

This result means that we are doing policy improvement with approximate Q-values. Then, [20, Lemma 6.1] tells that doing policy improvement using approximate Q-values gives us approximate policy improvement:

$$\mathbb{E}_{\pi'(\tau|s_g)}\left[\sum_{t=0}^{\infty} \gamma^t r_{s_g}(s_t, a_t)\right] \geq \mathbb{E}_{\pi(\tau|s_g)}\left[\sum_{t=0}^{\infty} \gamma^t r_{s_g}(s_t, a_t)\right] - \frac{2\gamma\epsilon}{1-\gamma},$$

for all goals $s_g \in \{s_g \mid p_g(s_g) > 0\}$. □

B.4 CONTRASTIVE RL (CPC)

In this section, we derive a version of contrastive RL based on the infoNCE objective [177]. 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. 3.7 with the following infoNCE objective [177]:

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

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 [153, 177, 187]

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

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 [241]:

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

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.

B.5 CONTRASTIVE RL (NCE + C-LEARNING)

In this section we describe `contrastive RL (NCE + C-learning)` the combined NCE + C-learning method used in Sec. 3.5.3 (Fig. 3.5). Math-

ematically, the NCE + C-learning objective is a simple, unweighted sum of the C-learning objective and the NCE objective:

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

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 $1+1-\gamma = 2-\gamma$:

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

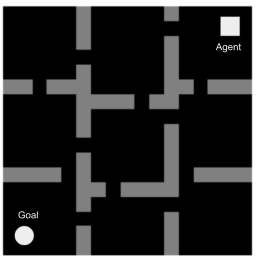
This trick increases the effective sample size by 96% (130 → 256, as measured using [116]).

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

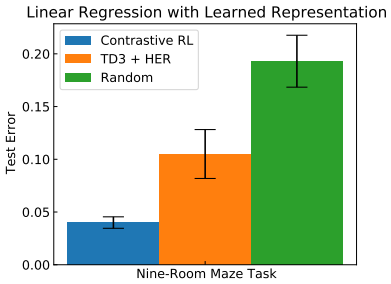
$$\begin{aligned}\mathcal{L}_2(f) \triangleq& \gamma\mathbb{E}_{\substack{s_g\sim p_g(s_g),(s_t,a_t)\sim p(s,a),\\ s_{t+1}\sim p(s_{t+1}|s_t,a_t),a_{t+1}\sim\pi(a_{t+1}|s_{t+1},s_g)}}\Big[\lfloor\exp(f(s_{t+1},a_{t+1},s_g))\rfloor_{\text{sg}}\log\sigma(f(s,a,s_f=s_g))\Big] \\ &+ \mathbb{E}_{s_g\sim p_g(s_g),(s,a)\sim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

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. 3.4.4 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 $\mathcal{L}(f) = \mathcal{L}_1(f) + \mathcal{L}_2(f)$.

B.6 ADDITIONAL EXPERIMENTS



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



(b) **Linear probe experiment.**

Figure b.2: **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.

B.6.1 *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 [5, 93] 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. b.2a 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. b.2b shows that features learned by contrastive RL can predict this distance better than all baselines.

As shown in Fig. b.2b, contrastive RL (NCE) learns representations that achieve lower test error than those learned by TD3+HER and by a random CNN encoder.

B.6.2 *When is contrastive learning better than learning a foreward model?*

In Fig. 3.3a, 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

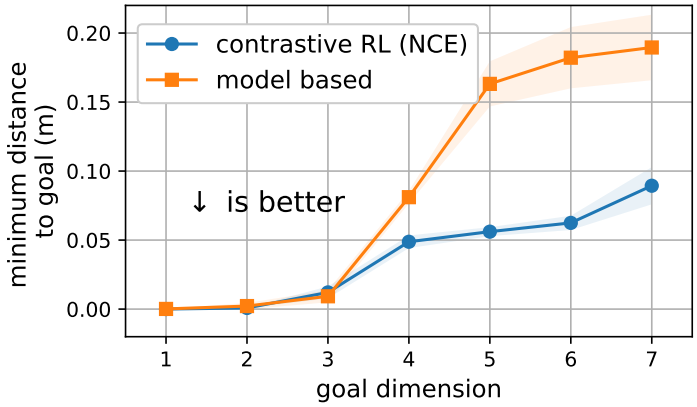


Figure b.3: Contrasive learning outperforms a forward model when the goal is 4-dimensional or larger. Error bars show the standard deviation across 5 random seeds.

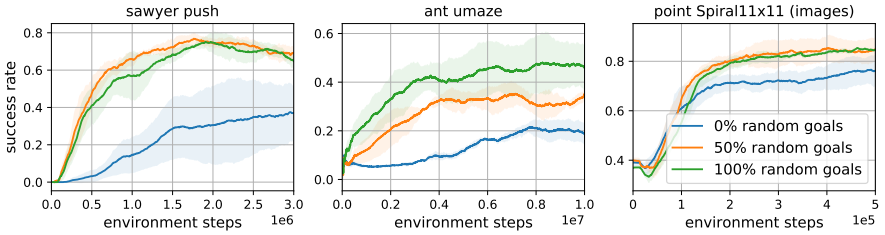


Figure b.4: **Goals used for the actor loss.** Goals are either sampled from the distribution over future states or from a distribution of random states. Error bars show the standard deviation across 5 random seeds.

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. b.3. 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 experiment provides some evidence that contrastive RL may be preferred over a forward model, even for tasks with very low dimensional goals.

B.6.3 *Goals used in the actor loss*

In theory, the distribution of goals for the actor loss (Eq. 3.8) 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

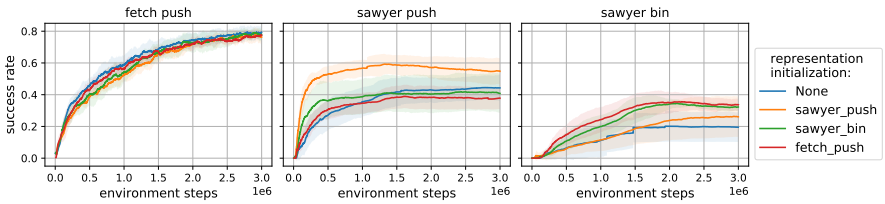


Figure B.5: **Transferring representations to solve new tasks.** After training the representations on one task for 1M environment steps, we used them to initialize a new agent for solving a new task.

ablation experiment to study this decision, and show results in Fig. B.4. 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.

B.6.4 *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 [265] 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. B.5. 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.

B.6.5 *Robustness to Environment Perturbations*

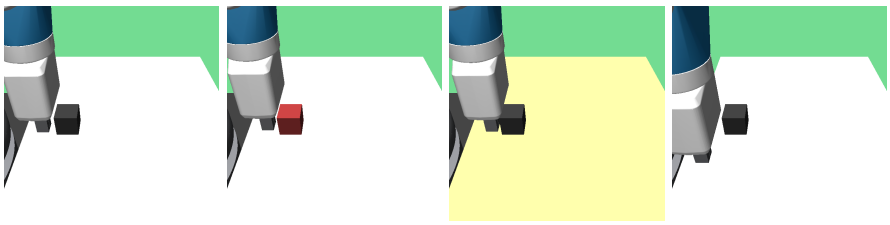


Figure b.6: Perturbations to the image-based `fetch push` environment.

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. b.6):

- 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.

B.6.6 *Additional figures*

This section presents additional figures.

- Fig. b.7 compares contrastive RL (NCE) with varying values of the filtering parameter ϵ, described in Sec. 3.4.5.
- Fig. b.8 – 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. b.9 – 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 [117] to rectify the embeddings to a grid.

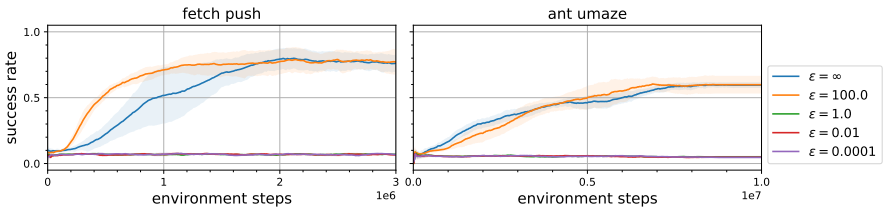


Figure b.7: **Filtered relabeling.** We filter the relabeled experience so that the agent only trains on experience where the probability under the commanded goal is similar to the probability under the actually-reached goal. While such filtering is required to prove convergence, these results suggest that good performance can be achieved without this filtering step.

- Fig. b.10 – A TSNE embedding of image representations from the `point Spiral11x11` task.
- Fig. b.11 – 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, $\langle \frac{\partial f}{\partial s}|_{(s,g)}, \frac{\partial f}{\partial s}|_{(s,g')} \rangle$.

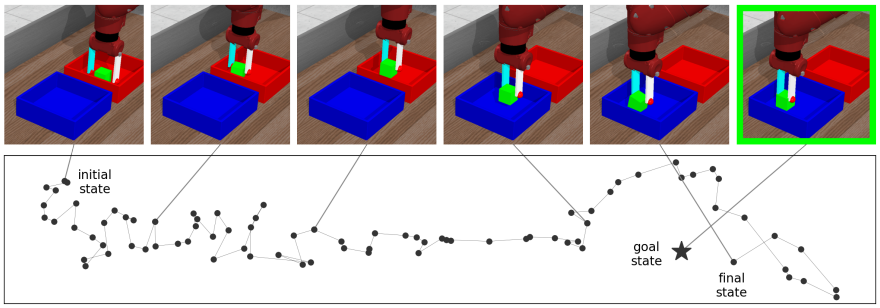


Figure b.8: **Visualizing the learned representations.** *(Top)* We show five observations from the bin picking task, as well as the goal image. *(Bottom)* A TSNE embedding of the image representations $\phi(s, a)$ learned by Contrastive RL (NCE). Note that different parts of the task (e.g., reaching, picking, placing) are well separated in the learned representation space.

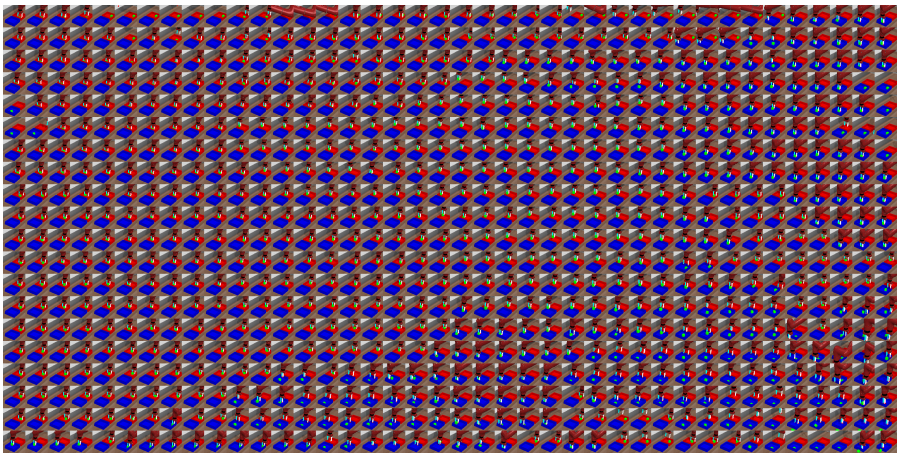
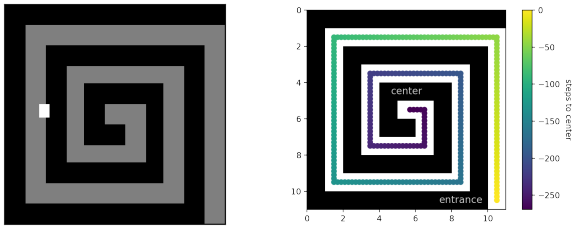
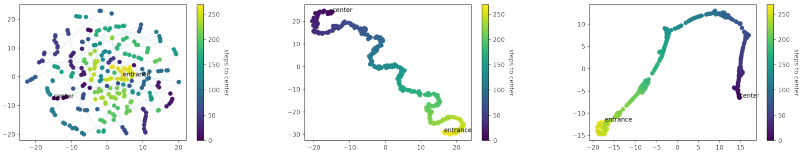


Figure b.9: Visualizing the image representations learned by our method on the `sawyer bin`. We compute a TSNE embedding of the representations, and then align the embeddings to a grid using RasterFairy [117].

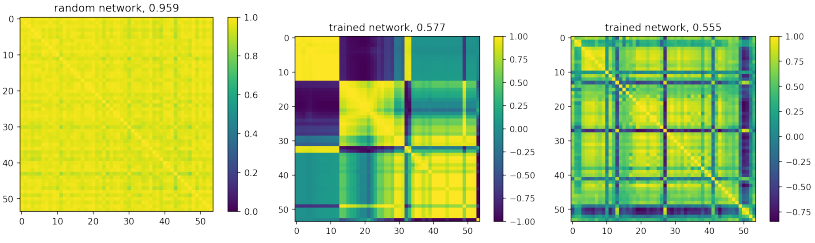


(a) (b)



(c) untrained encoder (d) contrastive RL (NCE) (e) TD3 + HER

Figure b.10: **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 B.11: **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, 122, 259, 264]. 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.

C

SEARCH ON THE REPLAY BUFFER

C.1 EFFICIENT SHORTEST PATH COMPUTATION

Our policy solves a shortest path problem every time it recomputes a new waypoint. Naïvely running Dijkstra's algorithm to compute a shortest path among the states in our active set $\mathcal{B}$ requires $O(|\mathcal{B}|^2)$ queries of our value function. While the search algorithm itself is fast, it is expensive to evaluate the value function on each pair of states at every time step. In our implementation (Algorithm 7), we amortize this computation across many calls to the policy. We periodically periodically evaluate the value function on each pair of nodes in the replay buffer, and then used the Floyd Warshall algorithm to compute the shortest path between all pairs. This takes $O(|\mathcal{B}|^3)$ time, but only $O(|\mathcal{B}|^2)$ calls to the value function. Let $D \in \mathbb{R}^{|\mathcal{B}|\times|\mathcal{B}|}$ be the resulting matrix storing the shortest path distances between all pairs of states in the active set. Now, given a start state s and goal state g, the shortest path distance is

$$d_{\text{sp}}(s,g) = \min\left(\min_{u,v\in\mathcal{T}} d(s,u) + D[u,v] + d(v,g), d(s,g)\right)$$

This computation requires $O(|\mathcal{B}|)$ calls to the value function, substantially better than the $O(|\mathcal{B}|^2)$ calls required with the naïve implementation.

C.2 ENVIRONMENTS

We used two simple navigation environments, Point-U and Point-FourRooms, shown in Figure 4.3a. In both environments, the observations are the location of the agent, $s = (x,y) \in \mathbb{R}^2$. The agent's actions $a = (dx,dy) \in [-1,1]^2$ are added to the agents current position at every time step. We tuned the environments so that the goal-conditioned algorithm (which we will use as a baseline) would perform as well as possible. Observing that the agent would get stuck at corners, we modified the environment to automatically add Gaussian noise to the agents action. The resulting dynamics were

$$s_{t+1} = \texttt{proj}(s_t + a_t + \epsilon_t) \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0,\sigma^2)$$

where `proj()` handles collisions with walls by projecting the state to the nearest free state. We used $\sigma^2 = 1.0$ for Point-U, and $\sigma^2 = 0.1$ for the (larger) Point-FourRooms environment.

Algorithm 7 Inputs are the current state s, the goal state g, the replay buffer $\mathcal{B}$, and the value function V. Returns the length and first waypoint of the shortest path.

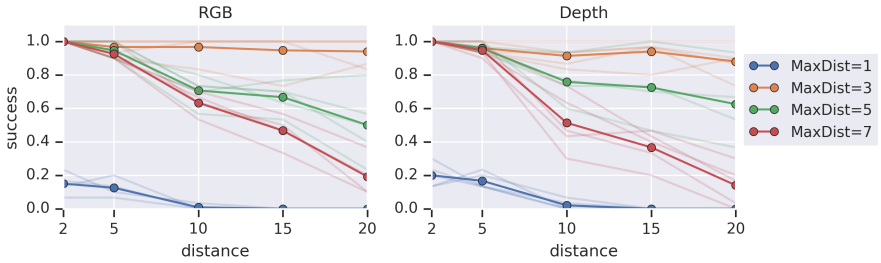
function ShortestPath($s, s_g, \mathcal{B}, V$)
  // Matrices: $D_\pi, D_{\mathcal{B}\to\mathcal{B}}, D_{s\to s_g} \in \mathbb{R}^{|\mathcal{B}|\times|\mathcal{B}|}$
  // Vectors: $D_{s\to\mathcal{B}}, D_{\mathcal{B}\to g} \in \mathbb{R}^{|\mathcal{B}|}$
  $D_\pi \leftarrow -V(\mathcal{B}, \mathcal{B})$ ▷ cached
  $D_{\mathcal{B}\to\mathcal{B}} \leftarrow$ FloydWarshall(D_π) ▷ cached
  $D_{s\to\mathcal{B}} \leftarrow -V(s, \mathcal{B})$
  $D_{\mathcal{B}\to g} \leftarrow -V(\mathcal{B}, g)$
  $D_{s\to g} \leftarrow D_{s\to\mathcal{B}} + D_{\mathcal{B}\to\mathcal{B}} + (D_{\mathcal{B}\to g})^T$
  $s_{w_1} \leftarrow \arg\min_{u,v\in\mathcal{B}} D_{s\to g}$
  return s_{w_1}

C.2.1 *Visual Navigation*

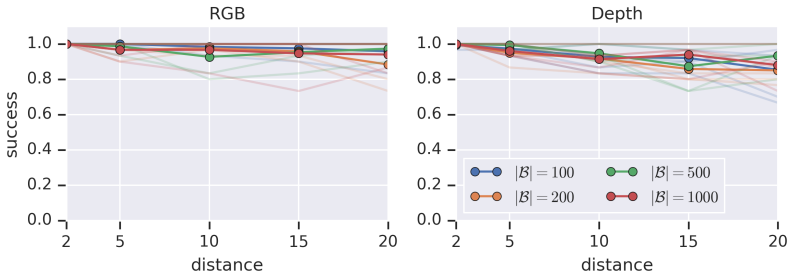
We ran most experiments on SUNCG house `0bda523d58df2ce52d0a1d90ba21f95c`. We repeated all experiments on SUNCG house `0601a680273d980b791505cab993096a`, with nearly identical results. We manually choose houses using the following criteria (1) single story, (2) no humans, and (3) included multiple rooms to make planning challenging. During training, we sampled "nearby" goal states (within 4 steps) for 80% of episodes, and sampled goals uniformly at random for the remaining 20% of episodes. We tuned these parameters to make goal-conditioned RL work as well as possible. We implemented goal-relabelling [10, 110], choosing between the (1) originally sampled goal, the (2) current state, and (3) a future state in the same trajectory, each with probability 33%. The agent's actions space was to move North/South/East/West. Observations were panoramic images, created by concatenating the first-person views from each of the cardinal directions. We used ensembles of 3 value functions, each with entirely independent weights. For all neural networks conditioned on both the current observation and the goal observation, we concatenated the current observation and goal observation along their last channel. For RGB images, this resulted in an input with dimensions $H \times W \times 6$. For depth images, the concatenated input had dimension $H \times W \times 2$.

C.3 ABLATION EXPERIMENTS

Because SoRB plans over a fixed replay buffer, one potential concern is that performance might degrade if the replay buffer is too small. To test this concern, we ran an experiment varying the size of the replay buffer. As shown in Figure C.1b, decreasing the replay buffer by a factor of 10x led to no discernible drop on performance. While we do expect



(a) Maximum edge length



(b) Replay buffer size

Figure c.1: **Sensitivity to Hyperparameters**: *(Top)* When constructing our graph, we ignore edges that are longer than some distance, MaxDist. We find that this hyperparameter is important to the success of our method. *(Bottom)* While we used a buffer of 1000 observations for most of our experiments, decreasing the buffer size has little effect on the method's success rate.

performance to drop if we further decrease the size of the replay buffer, the requirement of storing 100 states (even high-resolution images) seems relatively minor. In a second ablation experiment, we varied the MaxDist hyperparameter that governs when we stop adding new edges to the graph. As shown in Figure c.1a, SoRB is sensitive to this hyperparameter, with values too large and too smaller leading to worse performance. When the MaxDist parameter is too small, graph search fails to find a path to the goal state. As we increase MaxDist, we increase the probability of underestimating the distance between pairs of states. We expect that improvements in uncertainty quantification in RL will improve the stability of our method w.r.t. this hyperparameter.

d

JOINT MODEL-POLICY OPTIMIZATION FOR MODEL-BASED RL

D.1 PROOFS AND ADDITIONAL ANALYSIS

D.1.1 *VMBPO Maximizes an Upper Bound on Return*

While MnM aims to maximize the (log) of the expected return, VMBPO aims to maximize the expected *exponentiated* return:

$$\text{MnM:} \quad \log \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \qquad \text{VMBPO:} \quad \log \mathbb{E}_\pi \left[e^{\eta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)} \right],$$

where $\eta > 0$ is a temperature term used by VMBPO. Note that maximizing the log of the expected return, as done by MnM, is equivalent to maximizing the expected return, as the function $\log(\cdot)$ is monotone increasing. However, maximizing the log of the expected *exponentiated* return, as done by VMBPO, is not equivalent to maximizing the expected return. Rather, it corresponds to maximizing a sum of the expected return and the *variance* of the return [155, Page 272]:

$$\frac{1}{\eta} \log \mathbb{E}_\pi \left[e^{\eta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)} \right] = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] + \frac{\eta}{2} \text{Var}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] + \mathcal{O}(\eta^2).$$

Thus, in environments with stochastic dynamics or rewards (e.g., the didactic example in Fig. 5.3), VMBPO will prefer to receive lower returns if the variance of the returns is much higher. We note that the expected *exponentiated* return is an *upper* bound on the expected return:

$$\log \mathbb{E}_\pi \left[e^{\eta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)} \right] \geq \eta \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].$$

This statement is a direct application of Jensen's inequality. The bound holds with a strict inequality in almost all MDPs. The one exception is trivial MDPs where all trajectories have exactly the same return. Of course, even a random policy is optimal for these trivial MDPs.

D.1.2 *Helper Lemmas*

We start by introducing a simple identity that will help handle discount factors in our analysis.

Lemma 10. *Define $p(H) = \text{GEOM}(1-\gamma)$ as the geometric distribution. Let discount factor $\gamma \in (0,1)$ and random variable x_t be given. Then the following identity holds:*

$$\mathbb{E}_{p(H)}\left[\sum_{t=0}^{H} x_t\right] = \sum_{t=0}^{\infty} \gamma^t x_t.$$

The proof involves substituting the definition of the Geometric distribution and then rearranging terms.

Proof.

$$\begin{aligned}
\mathbb{E}_{p(H)}\left[\sum_{t=0}^{H} x_t\right] &= (1-\gamma)\sum_{H=0}^{\infty}\gamma^H \sum_{t=0}^{H} x_t \\
&= (1-\gamma)\left(x_0 + \gamma(x_0+x_1) + \gamma^2(x_0+x_1+x_2) + \cdots\right) \\
&= (1-\gamma)\left(x_0(1+\gamma+\gamma^2+\cdots) + x_1(\gamma+\gamma^2+\cdots) + \cdots\right) \\
&= (1-\gamma)\left(x_0\frac{1}{1-\gamma} + x_1\frac{\gamma}{1-\gamma} + x_2\frac{\gamma^2}{1-\gamma} + \cdots\right) \\
&= \sum_{t=0}^{\infty}\gamma^t x_t.
\end{aligned}$$

□

The second helper lemma describes how the discounted expected return objective can be written as the expected *terminal* reward of a mixture of finite-length episodes.

Lemma 11. *Define $p(H) = \text{GEOM}(1-\gamma)$ as the geometric distribution, and $p(\tau \mid H)$ as a distribution over length-H episodes. We can then write the expected discounted return objective as follows:*

$$\begin{aligned}
\mathbb{E}_{p(\tau|H=\infty)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] &= \frac{1}{1-\gamma}\mathbb{E}_{p(H)}\left[\mathbb{E}_{p(\tau|H=H)}\left[r(s_H,a_H)\right]\right] \\
&= \frac{1}{1-\gamma}\iint p(H)p(\tau \mid H=H)r(s_H,a_H)d\tau dH.
\end{aligned}$$

Proof. The first identity follows from the definition of the geometric distribution. The second identity writes the expectations as integrals, which will make future analysis clearer. □

D.1.3 *Proof of Lemma 3*

Proof.

$$
\begin{aligned}
&\log \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \overset{(a)}{=} \log\frac{1}{1-\gamma}\iint p(H)p(\tau \mid H=H)r(s_H,a_H)d\tau dH \\
&= \log\iint p(H)\frac{p(\tau \mid H=H)}{q_\theta(\tau \mid H=H)}q_\theta(\tau \mid H=H)r(s_H,a_H)d\tau dH - \log(1-\gamma) \\
&\overset{(b)}{\geq} \int p(H)\left(\log\int \frac{p(\tau \mid H=H)}{q_\theta(\tau \mid H=H)}q_\theta(\tau \mid H=H)r(s_H,a_H)d\tau\right)dH - \log(1-\gamma) \\
&\overset{(c)}{\geq} \iint p(H)q_\theta(\tau \mid H=H)\left(\log p(\tau \mid H=H) - \log q_\theta(\tau \mid H=H) + \log r(s_H,a_H)\right)d\tau dH - \log(1-\gamma) \\
&\overset{(d)}{=} \iint p(H)q_\theta(\tau \mid H=H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) + \cancel{\log \pi_\theta(a_t\mid s_t)} - \log q_\theta(s_{t+1}\mid s_t,a_t) - \cancel{\log \pi_\theta(a_t\mid s_t)}\right) + \log r(s_H,a_H)\right)d\tau dH \\
&\quad - \log(1-\gamma) \\
&\overset{(e)}{=} \iint p(H)q_\theta(\tau \mid H=\infty)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log r(s_H,a_H)\right)d\tau dH - \log(1-\gamma) \\
&\overset{(f)}{=} \int q_\theta(\tau)\int p(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log r(s_H,a_H)\right)dHd\tau - \log(1-\gamma) \\
&\overset{(g)}{=} \int q_\theta(\tau)\mathbb{E}_{p(H)}\left[\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log r(s_H,a_H)\right]d\tau - \log(1-\gamma) \\
&\overset{(h)}{=} \int q_\theta(\tau)\sum_{t=0}^{\infty}\gamma^t\left(\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t) + (1-\gamma)\log r(s_H,a_H)\right)d\tau - \log(1-\gamma) \\
&\overset{(i)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t\left(\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t) + (1-\gamma)\log r(s_H,a_H) - (1-\gamma)\log(1-\gamma)\right)\right].
\end{aligned}
$$

For *(a)*, we applied Lemma 11. For *(b)*, we applied Jensen's inequality. For *(c)*, we applied Jensen's inequality again. For *(d)*, we substituted the definitions of $p_\theta(\tau \mid H)$ and $q_\theta(\tau \mid H)$. For *(e)*, we noted that the term inside the summation only depends on the first H steps of the trajectory, so collecting longer trajectories will not change the result. This allows us to rewrite the integral as an expectation using a single infinite-length trajectory. For *(f)*, we recalled the definition $q_\theta(\tau) = q_\theta(\tau = H = \infty)$ and swap the order of integration. For *(g)*, we express the inner integral over $p(H)$ as an expectation. For *(h)*, we applied the identity from Lemma 10. For *(i)*, we moved the constant $\log(1-\gamma)$ back inside the integral and rewrote the integral as an expectation. We have thus obtained the desired result. □

D.1.4 *Proof of Lemma 4*

Before presenting the proof of Lemma 3 itself, we show how we derived the lower bound in this more general case. While this step is not required for the proof, we include it because it sheds light on how similar lower bounds might be derived for other problems. We define

$\gamma_\theta(H)$ to be a learned distribution over horizons H. We then proceed, following many of the same steps as for the proof of Lemma 3.

$$\log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \overset{(a)}{=} \log \iint \frac{p(\tau,H)}{q_\theta(\tau,H)} q_\theta(\tau,H) r(s_H,a_H) d\tau dH - \log(1-\gamma)$$

$$\overset{(b)}{\geq} \iint q_\theta(\tau,H)\left(\log p(\tau,H) - \log q_\theta(\tau,H) + \log r(s_H,a_H)d\tau\right) dH - \log(1-\gamma) \tag{d.1}$$

$$\overset{(c)}{=} \int \sum_{H=0}^{\infty} \gamma_\theta(H) q_\theta(\tau \mid H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log p(H) - \log\gamma_\theta(H) + \log r(s_H,a_H)d\tau\right) - \log(1-\gamma)$$

$$\overset{(d)}{=} \int q_\theta(\tau \mid H=\infty) \sum_{H=0}^{\infty} \gamma_\theta(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log p(H) - \log\gamma_\theta(H) + \log r(s_H,a_H)d\tau\right) - \log(1-\gamma)$$

$$\overset{(e)}{=} \int q_\theta(\tau) \sum_{H=0}^{\infty} \gamma_\theta(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log p(H) - \log\gamma_\theta(H) + \log r(s_H,a_H)d\tau\right) - \log(1-\gamma)$$

$$\overset{(f)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty} \gamma_\theta(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \cancel{\log(1-\gamma)} + H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right] - \cancel{\log(1-\gamma)}$$

$$\overset{(g)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\left(\sum_{t=H}^{\infty} q(t)\right)\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + \gamma_\theta(H)\left(H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right]$$

$$\overset{(h)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\left(1 - \sum_{t=0}^{H-1} q(t)\right)\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + \gamma_\theta(H)\left(H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right]$$

$$\overset{(i)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\left(1 - \Gamma_\theta(H-1)\right)\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + \gamma_\theta(H)\left(H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right]$$

$$\overset{(j)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\gamma_\theta(H)\left(\frac{1-\Gamma_\theta(H-1)}{\gamma_\theta(H)}\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right].$$

For *(a)*, we applied Lemma 11 and multiplied the integrand by $\frac{q_\theta(\tau|H=H)\gamma_\theta(H)}{q_\theta(\tau|H=H)\gamma_\theta(H)} = 1$. For *(b)*, we applied Jensen's inequality. For *(c)*, we factored $p(\tau,H) = p(\tau,H)p(H)$ and $q_\theta(\tau,H) = q(\tau \mid H)\gamma_\theta(H)$. Note that under the joint distribution $p(\tau,H)$, the horizon $H \sim p(H) = \textsc{Geom}(1-\gamma)$ is independent of the trajectory, τ. For *(d)*, we rewrote the expectation as an expectation over a single infinite-length trajectory and simplified the summand. For *(e)*, we recall the definition $q_\theta(\tau) = q_\theta(\tau = H = \infty)$. For *(f)*, we rewrote the integral as an expectation and wrote out the definition of the geometric distribution, $p(H)$. For *(g)*, we regrouped the difference of dynamics terms. For *(h)*, we noted used the fact that $\sum_{t=0}^{H-1}\gamma_\theta(t) + \gamma_{t=H}^{\infty}\gamma_\theta(t) = 1$. For *(i)*, we substituted the definition of the CDF function. For *(j)*, we rearranged terms so that all were multiplied by the discount $\gamma_\theta(H)$. Thus, we have obtained the desired result. We now prove Lemma 4, showing that Eq. 5.4 becomes tight at optimality.

Proof.

$$\mathcal{L}_\gamma(\theta) \overset{(a)}{=} \iint q_\theta(\tau,H)\left(\log p(\tau,H) - \log q_\theta(\tau,H) + \log r(s_H,a_H)d\tau\right) dH - \log(1-\gamma)$$

$$\overset{(b)}{=} \iint q_\theta(\tau)\gamma_\theta(H\mid\tau)\left(\log p(\tau) + \log p(H) - \log q_\theta(\tau) - \log\gamma_\theta(H\mid\tau) + \log r(s_H,a_H)d\tau\right) dH - \log(1-\gamma) \tag{d.2}$$

For *(a)*, we undo some of the simplifications above, going back to Eq. d.1 For *(b)*, we factor $q_\theta(\tau,H) = q_\theta(\tau)\gamma_\theta(H \mid \tau)$ and $p(\tau,H) =$

$p(\tau)p(H)$. At this point, we can solve analytically for the optimal discount distribution, $\gamma_\theta(H \mid \tau)$:

$$\gamma^*_\theta(H \mid \tau) = \frac{p(H)r(s_H, a_H)}{\sum_{H'=0}^{\infty} p(H')r(s_{H'}, a_{H'})} = \frac{p(H)r(s_H, a_H)}{(1-\gamma)R(\tau)} \tag{d.3}$$

In the second equality, we substitute the definition of $R(\tau)$. We then substitute Eq. d.3 into our expression for $\mathcal{L}_\gamma(\theta)$ and simplify the resulting expression.

$$\begin{aligned}
\mathcal{L}_\gamma(\theta) &= \iint q_\theta(\tau)\gamma_\theta(H \mid \tau)\,(\log p(\tau) + \cancel{\log p(H)} - \log q_\theta(\tau) - \cancel{\log p(H)} - \cancel{\log r(s_H, a_H)} \\
&\qquad + \cancel{\log(1-\gamma)} + \log R(\tau) + \cancel{\log r(s_H, a_H)} d\tau)\, dH - \cancel{\log(1-\gamma)} \\
&= \iint q_\theta(\tau)\gamma_\theta(H \mid \tau)\,(\log p(\tau) - \log q_\theta(\tau) + \log R(\tau) d\tau)\, dH \\
&= \int q_\theta(\tau)\,(\log p(\tau) - \log q_\theta(\tau) + \log R(\tau))\, d\tau.
\end{aligned}$$

In the final line we have removed the integral over H because none of the integrands depend on H. At this point, we can solve analytically for the optimal trajectory distribution, $q_\theta(\tau)$:

$$q^*(\tau) = \frac{p(\tau)R(\tau)}{\int p(\tau')R(\tau')d\tau'}. \tag{d.4}$$

We then substitute Eq. d.4 into our expression for $\mathcal{L}_\gamma(\theta)$, and simplify the resulting expression:

$$\begin{aligned}
\mathcal{L}_\gamma(\theta) &= \int q_\theta(\tau)\left(\cancel{\log p(\tau)} - \cancel{\log p(\tau)} - \cancel{\log R(\tau)} + \log \int p(\tau')R(\tau')d\tau' + \cancel{\log R(\tau)}\right) d\tau \\
&= \log \int p(\tau)R(\tau)d\tau = \log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].
\end{aligned}$$

We have thus shown that the lower bound $\mathcal{L}_\gamma$ becomes tight when we use the optimal distribution over trajectories $q_\theta(\tau)$ and optimal learned discount $\gamma_\theta(H \mid \tau)$. □

D.1.5 *A lower bound for goal-reaching tasks.*

Many RL problems can be better formulated as goal-reaching problems, a formulation that does not require defining a reward function. We now introduce a variant of our method for goal-reaching tasks. Using $\rho^\pi(s_{t+})$ to denote the discounted state occupancy measure of policy π, we define the goal-reaching objective as maximizing the probability density of reaching a desired goal s_g:

$$\max_\theta \log \rho^{\pi_\theta}(s_{t+} = s_g). \tag{d.5}$$

We refer the reader to [57] for a more detailed discussion of this objective. For simplicity, we assume that the goal is fixed, noting that the multi-task setting can be handled by conditioning the policy on

the commanded goal. Similar to Lemma 3, we can construct a lower bound on the goal-conditioned RL problem:

Lemma 12. *Let initial state distribution $p_1(s_1)$, real dynamics $p(s_{t+1} \mid s_t, a_t)$, reward function $r(s_t, a_t) > 0$, discount factor $\gamma \in (0, 1)$, and goal g be given. Then the following bound holds for* any *dynamics $q(s_{t+1} \mid s_t, a_t)$ and policy $\pi(a_t \mid s_t)$:*

$$\log p^{\pi_\theta}(s_{t+} = s_g) \geq \mathbb{E}_{q^{\pi_\theta}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right],$$

where $\tilde{r}_g(s_t, a_t, s_{t+1}) \triangleq (1-\gamma)\log p(s_{t+1} = s_g \mid s_t, a_t) + \log p(s_{t+1} \mid s_t, a_t) - \log q(s_{t+1} \mid s_t, a_t)$.

The proof, presented below, is similar to the proof of Lemma 3. The first term in the reward function, the log probability of reaching the commanded goal one time step in the future, is similar to prior work [198]. The correction term $\log p - \log q$ incentivizes the policy to avoid transitions where the model is inaccurate, and can be estimated using a separate classifier. One important aspect of this goal-reaching problem is that it is entirely data-driven, avoiding the need for any manually-designed reward functions.

Proof.

$$\begin{aligned}
&\log \rho^{\pi_\theta}(s_{t+} = s_g) \overset{(a)}{=} \log \iint p(H)p(\tau \mid H = H)p(s_g \mid s_H, a_H)d\tau dH \\
&= \log \iint p(H)\frac{p(\tau \mid H = H)}{q_\theta(\tau \mid H = H)} q_\theta(\tau \mid H = H)\frac{p(s_g \mid s_H, a_H)}{q_\theta(s_g \mid s_H, a_H)} q_\theta(s_g \mid s_H, a_H)d\tau dH - \log(1-\gamma) \\
&\overset{(c)}{\geq} \iint p(H)q_\theta(\tau \mid H = H)\left(\log p(\tau \mid H = H) - \log q_\theta(\tau \mid H = H) + \log p(s_g \mid s_H, a_H) - \log q_\theta(s_g \mid s_H, a_H)\right)d\tau dH - \log(1-\gamma) \\
&\overset{(d)}{=} \iint p(H)q_\theta(\tau \mid H = \infty)\left(\sum_{t=0}^{H} \log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t)\right) + \log p(s_g \mid s_H, a_H) - \log q_\theta(s_g \mid s_H, a_H)d\tau dH - \log(1-\gamma) \\
&\overset{(d)}{=} \int q_\theta(\tau)\int p(H)\left(\sum_{t=0}^{H} \log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t)\right) + \log p(s_g \mid s_H, a_H) - \log q_\theta(s_g \mid s_H, a_H)dHd\tau - \log(1-\gamma) \\
&\overset{(d)}{=} \int q_\theta(\tau)\sum_{t=0}^{\infty}\gamma^t\left(\log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t) + (1-\gamma)(\log p(s_g \mid s_t, a_t) - \log q_\theta(s_g \mid s_t, a_t)\right)d\tau - \log(1-\gamma) \\
&\overset{(d)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t\left(\log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t) + (1-\gamma)(\log p(s_g \mid s_t, a_t) - \log q_\theta(s_g \mid s_t, a_t) + (1-\gamma)\log(1-\gamma)\right)\right].
\end{aligned}$$

□

Similar to the more complex lower bound presented in Eq. 5.4, this lower bound on goal-reaching can be modified (by learning a discount factor) to become a tight lower bound. The resulting objective would resemble a model-based version of the algorithm from Rudner et al. [198].

D.1.6 *Derivation of Model Objective (Eq. 5.10)*

Our lower bound depends on entirely trajectories sampled from the learned dynamics. In this section, we show how the same objective can be expressed as an expectation of transitions. This expression is

easier to optimize, as it does not require backpropagating gradients through time. We start by writing our lower bound, conditioned on a current state s_t.

$$\begin{aligned}
&\mathbb{E}_{\substack{\pi(a_t|s_t),\\ q_\theta(s_{t+1}|s_t,a_t)}}\left[\sum_{t'=t}^{\infty}\gamma^{t'-t}\tilde{r}(s_{t'},a_{t'})\mid s_t\right] \\
&\qquad = \mathbb{E}_{\substack{\pi(a_t|s_t),\\ q_\theta(s_{t+1}|s_t,a_t)}}\left[\tilde{r}(s_t,a_t,s_{t+1})+\gamma V(s_{t+1})\mid s_t\right] \\
&\qquad \overset{(a)}{=} \mathbb{E}_{\substack{\pi(a_t|s_t),\\ q_\theta(s_{t+1}|s_t,a_t)}}\left[(1-\gamma)\log r(s_t,a_t)+\log\frac{C_\phi(s_t,a_t,s_{t+1})}{1-C_\phi(s_t,a_t,s_{t+1})}\right. \\
&\qquad\qquad \left.-(1-\gamma)\log(1-\gamma)+\gamma V(s_{t+1})\mid s_t\right]
\end{aligned}$$

In *(a)*, we substituted the definition of the augmented return. For the purpose of optimizing the dynamics model, we can ignore all terms that do not depend on s_{t+1}. Removing these terms, we arrive at our model training objective (Eq. 5.10)

D.2 ADDITIONAL EXPERIMENTS

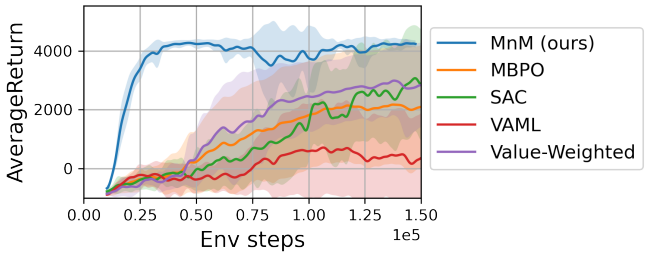


Figure d.1: **Alternative Model Learning Objectives**: Using the `DClawScrewFixed-v0` task, we compare MnM and MBPO [105] to two additional model learning objectives suggested in the literature, VAML [61] and value-weighted maximum likelihood [127]. MnM (our method) outperforms these alternative approaches.

We compare MnM to a number of alternative model learning methods. MBPO [105] uses a standard maximum likelihood model. We implement a version of VAML [61], which augments the maximum likelihood loss with an additional temporal difference loss; the model should predict next states that have low Bellman error. Finally, we compare to a variant of the MBPO maximum likelihood model that weights transitions based on the Q values, an idea discussed (but not actually implemented) in Lambert et al. [127]. We implement this value weighting method by computing the Q values for the current states and computing a softmax over the batch dimension to obtain per-example weights.

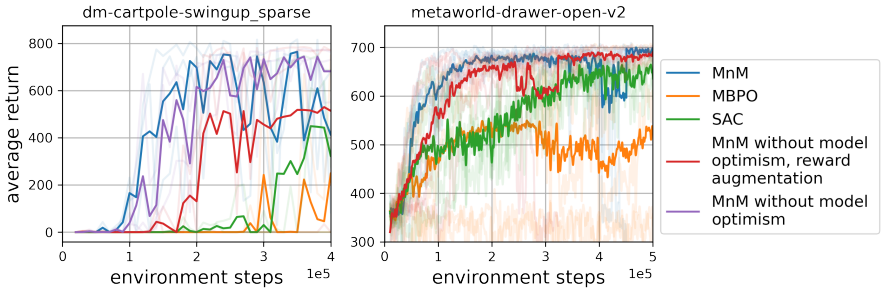


Figure d.2: **Ablation Experiments**: Compared with MBPO (orange line), MnM uses a GAN-like model (red line) with a model optimism term and modifies the reward function.

We use the `DClawScrewFixed-v0` task for this experiment. The results, shown in Fig. d.1, show that MnM outperforms these alternative approaches. We observe that the value-weighting performs slightly better than the standard maximum likelihood model, while the VAML method performs noticeable worse.

We next run an ablation experiment to study the importance of a few key design decisions. We compare MnM with ablations that omit the reward augmentation and the model optimism term. The results shown in Fig. d.2 indicate that most of the benefit of MnM comes from using a GAN-like model. Because the dynamics of these tasks are nearly deterministic, it is not surprising that the optimistic dynamics and the reward augmentation have only a small effect.

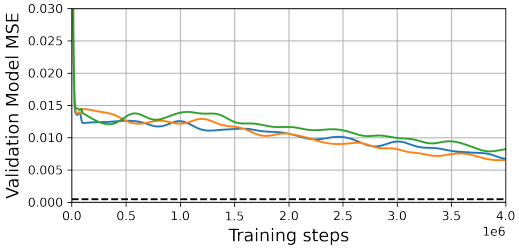


Figure d.3: **MnM trains stably.** Despite resembling a GAN, the MnM dynamics model trains stably, with the validation MSE decreasing steadily throughout training. Different colors correspond to different random seeds of MnM. The dashed line corresponds to the minimum validation MSE of a maximum likelihood dynamics model.

With the implementation details described in [55], we found that the MnM dynamics model trained stably, despite resembling a GAN. In Fig. d.3, we plot the validation MSE of the MnM model throughout training, observing that it decreases monotonically. Different lines correspond to different random seeds, and the dashed line corresponds to the minimum MSE of a maximum likelihood model (a MBPO model). Note that the MnM model is not trained with this MSE objective, but with the GAN-like objective in Eq. 5.10. It is therefore

not surprising that MnM does not perform as well on this objective as the maximum likelihood model.

BIBLIOGRAPHY

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. "Variational option discovery algorithms." In: *arXiv preprint arXiv:1807.10299* (2018).

[2] Joshua Achiam, Ethan Knight, and Pieter Abbeel. "Towards characterizing divergence in deep Q-learning." In: *arXiv preprint arXiv:1903.08894* (2019).

[3] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. "Learning to poke by poking: Experiential learning of intuitive physics." In: *Advances in Neural Information Processing Systems*. 2016, pp. 5074–5082.

[4] Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. "ROBEL: RObotics BEnchmarks for Learning with low-cost robots." In: *Conference on Robot Learning*. PMLR. 2020, pp. 1300–1313.

[5] Guillaume Alain and Yoshua Bengio. "Understanding intermediate layers using linear classifier probes." In: *arXiv preprint arXiv:1610.01644* (2016).

[6] B Amos, I Rodriguez, J Sacks, B Boots, and Z Kolter. "Differentiable MPC for End-to-end Planning and Control." In: *Advances in neural information processing systems* (2018).

[7] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. "Unsupervised state representation learning in Atari." In: *Advances in Neural Information Processing Systems* 32 (2019).

[8] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3674–3683.

[9] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular multitask reinforcement learning with policy sketches." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 166–175.

[10] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. "Hindsight experience replay." In: *Advances in neural information processing systems*. 2017, pp. 5048–5058.

[11] Raghuram Mandyam Annasamy and Katia Sycara. "Towards better interpretability in deep Q-networks." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4561–4569.

[12] Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. "Combating the compounding-error problem with a multi-step model." In: *arXiv preprint arXiv:1905.13320* (2019).

[13] Kavosh Asadi, Dipendra Misra, and Michael Littman. "Lipschitz continuity in model-based reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2018, pp. 264–273.

[14] Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The option-critic architecture." In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[15] J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. "Solving uncertain Markov decision processes." In: (2001).

[16] Xueying Bai, Jian Guan, and Hongning Wang. "A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation." In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 10735–10746.

[17] André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Shibl Mourad, David Silver, Doina Precup, et al. "The option keyboard: Combining skills in reinforcement learning." In: *Advances in Neural Information Processing Systems*. 2019, pp. 13052–13062.

[18] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. "Successor features for transfer in reinforcement learning." In: *Advances in neural information processing systems*. 2017, pp. 4055–4065.

[19] Marc G Bellemare, Will Dabney, and Rémi Munos. "A Distributional Perspective on Reinforcement Learning." In: *International Conference on Machine Learning (ICML)*. 2017. URL: https://arxiv.org/pdf/1707.06887.pdf.

[20] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[21] Peter J Bickel, David A Freedman, et al. "Some asymptotic theory for the bootstrap." In: *The annals of statistics* 9.6 (1981), pp. 1196–1217.

[22] Steffen Bickel, Michael Brückner, and Tobias Scheffer. "Discriminative learning for differing training and test distributions." In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 81–88.

[23] Léonard Blier, Corentin Tallec, and Yann Ollivier. "Learning successor states and goal-dependent values: A mathematical viewpoint." In: *arXiv preprint arXiv:2101.07123* (2021).

[24] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. "Universal successor features approximators." In: *arXiv preprint arXiv:1812.07626* (2018).

[25] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018. URL: http://github.com/google/jax.

[26] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI gym." In: *arXiv preprint arXiv:1606.01540* (2016).

[27] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations." In: *International conference on machine learning*. PMLR. 2019, pp. 783–792.

[28] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. "Goal-Conditioned Reinforcement Learning with Imagined Subgoals." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1430–1440.

[29] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. "Neural Topological SLAM for Visual Navigation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12875–12884.

[30] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. "Decision transformer: Reinforcement learning via sequence modeling." In: *Advances in neural information processing systems* 34 (2021).

[31] Song Chen, Junpeng Jiang, Xiaofang Zhang, Jinjin Wu, and Gongzheng Lu. "GAN-Based Planning Model in Deep Reinforcement Learning." In: *International Conference on Artificial Neural Networks*. Springer. 2020, pp. 323–334.

[32] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations." In: *ArXiv* abs/2002.05709 (2020).

[33] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 15745–15753.

[34] Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. "Learning Navigation Behaviors End-to-End With AutoRL." In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2007–2014. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2899918.

[35] Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. "Variational Empowerment as Representation Learning for Goal-Conditioned Reinforcement Learning." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1953–1963.

[36] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[37] Yinlam Chow, Brandon Cui, MoonKyung Ryu, and Mohammad Ghavamzadeh. "Variational model-based policy optimization." In: *arXiv preprint arXiv:2006.05443* (2020).

[38] Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep reinforcement learning from human preferences." In: *arXiv preprint arXiv:1706.03741* (2017).

[39] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." In: *Advances in Neural Information Processing Systems*. 2018, pp. 4754–4765.

[40] Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. "Gradient-aware model-based policy search." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3801–3808.

[41] Peter Dayan. "Improving generalization for temporal difference learning: The successor representation." In: *Neural Computation* 5.4 (1993), pp. 613–624.

[42] Marc Peter Deisenroth and Carl Edward Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." In: *International Conference on Machine Learning (ICML)*. 2011, pp. 465–472.

[43] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. "Goal-conditioned imitation learning." In: *Advances in Neural Information Processing Systems*. 2019, pp. 15324–15335.

[44] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning." In: *arXiv preprint arXiv:1605.09782* (2016).

[45] Alexey Dosovitskiy and Vladlen Koltun. "Learning to act by predicting the future." In: *arXiv preprint arXiv:1611.01779* (2016).

[46] Chris Drummond. "Accelerating reinforcement learning by composing solutions of automatically identified subtasks." In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 59–104.

[47] Yilun Du, Chuang Gan, and Phillip Isola. "Curious representation learning for embodied intelligence." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10408–10417.

[48] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. "Adversarially learned inference." In: *arXiv preprint arXiv:1606.00704* (2016).

[49] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control." In: *arXiv preprint arXiv:1812.00568* (2018).

[50] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. "RvS: What is Essential for Offline RL via Supervised Learning?" In: *arXiv preprint arXiv:2112.10751* (2021).

[51] Ben Eysenbach, Sergey Levine, and Russ R Salakhutdinov. "Replacing rewards with examples: Example-based policy search via recursive classification." In: *Advances in Neural Information Processing Systems* 34 (2021).

[52] Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. "Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers." In: *International Conference on Learning Representations*. 2020.

[53] Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. "Rewriting History with Inverse RL: Hindsight Inference for Policy Improvement." In: *ArXiv* abs/2002.11089 (2020).

[54] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. "Diversity is All You Need: Learning Skills without a Reward Function." In: *International Conference on Learning Representations*. 2018.

[55] Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. "Mismatched No More: Joint Model-Policy Optimization for Model-Based RL." In: *Advances in Neural Information Processing Systems*. 2022.

[56] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. "Search on the Replay Buffer: Bridging Planning and Reinforcement Learning." In: *Advances in Neural Information Processing Systems* 32 (2019).

[57] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. "C-Learning: Learning to Achieve Goals via Recursive Classification." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=tc5qisoB-C.

[58] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. "C-Learning: Learning to Achieve Goals via Recursive Classification." In: *ArXiv* abs/2011.08909 (2021).

[59] Benjamin Eysenbach, Soumith Udatha, Sergey Levine, and Ruslan Salakhutdinov. "Imitating Past Successes can be Very Suboptimal." In: *arXiv preprint arXiv:2206.03378* (2022).

[60] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. "Curriculum-guided hindsight experience replay." In: *Advances in Neural Information Processing Systems*. 2019, pp. 12623–12634.

[61] Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. "Value-aware loss function for model-based reinforcement learning." In: *Artificial Intelligence and Statistics*. 2017, pp. 1486–1494.

[62] Aleksandra Faust, Oscar Ramirez, Marek Fiser, Ken Oslund, Anthony Francis, James Davidson, and Lydia Tapia. "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-based Planning." In: *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*. Brisbane, Australia, 2018, pp. 5113–5120. URL: https://arxiv.org/abs/1710.03937.

[63] Chelsea Finn and Sergey Levine. "Deep visual foresight for planning robot motion." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2786–2793.

[64] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. "Deep spatial autoencoders for visuomotor learning." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2016-June. IEEE. 2016, pp. 512–519. ISBN: 9781467380263. DOI: 10.1109/ICRA.2016.7487173. arXiv: 1509.06113.

[65] David Fischinger, Markus Vincze, and Yun Jiang. "Learning grasps for unknown objects in cluttered scenes." In: *2013 IEEE international conference on robotics and automation*. IEEE. 2013, pp. 609–616.

[66] Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. "Self-supervised Learning of Image Embedding for Continuous Control." In: *arXiv preprint arXiv:1901.00943* (2019).

[67] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. "Automatic goal generation for reinforcement learning agents." In: *International conference on machine learning*. PMLR. 2018, pp. 1515–1528.

[68] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. "Multi-level discovery of deep options." In: *arXiv preprint arXiv:1703.08294* (2017).

[69] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. "Meta learning shared hierarchies." In: *arXiv preprint arXiv:1710.09767* (2017).

[70] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. "D4rl: Datasets for deep data-driven reinforcement learning." In: *arXiv preprint arXiv:2004.07219* (2020).

[71] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adverserial Inverse Reinforcement Learning." In: *International Conference on Learning Representations*. 2018.

[72] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. "Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition." In: *NeurIPS*. 2018.

[73] Scott Fujimoto and Shixiang Shane Gu. "A minimalist approach to offline reinforcement learning." In: *Advances in Neural Information Processing Systems* 34 (2021).

[74] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods." In: *International Conference on Machine Learning (ICML)* (2018).

[75] Scott Fujimoto, Herke Van Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." In: *arXiv preprint arXiv:1802.09477* (2018).

[76] Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. "Learning to reach goals without reinforcement learning." In: *arXiv preprint arXiv:1912.06088* (2019).

[77] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. "Learning to Reach Goals via Iterated Supervised Learning." In: *International Conference on Learning Representations*. 2020.

[78] Raj Ghugare, Homanga Bharadhwaj, Benjamin Eysenbach, Sergey Levine, and Russ Salakhutdinov. "Simplifying Model-based RL: Learning Representations, Latent-space Models, and Policies with One Objective." In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=MQcmfgRxf7a.

[79] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems (NIPS)*. 2014. URL: https://arxiv.org/pdf/1406.2661.pdf.

[80] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. "Variational intrinsic control." In: *arXiv preprint arXiv:1611.07507* (2016).

[81] Jean-Bastien Grill et al. "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." In: *ArXiv* abs/2006.07733 (2020).

[82] Christopher Grimm, André Barreto, Satinder Singh, and David Silver. "The value equivalence principle for model-based reinforcement learning." In: *arXiv preprint arXiv:2011.03506* (2020).

[83] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. "Neural predictive belief representations." In: *arXiv preprint arXiv:1811.06407* (2018).

[84] Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. "Bootstrap latent-predictive representations for multitask reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3875–3886.

[85] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. "Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning." In: *arXiv preprint arXiv:1910.11956* (2019).

[86] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. "Cognitive mapping and planning for visual navigation." In: *arXiv preprint arXiv:1702.03920* 3 (2017).

[87] Michael U Gutmann and Aapo Hyvärinen. "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics." In: *Journal of machine learning research* 13.2 (2012).

[88] Michael Gutmann and Aapo Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 297–304.

[89] David Ha and Jürgen Schmidhuber. "World Models." In: *arXiv preprint arXiv:1803.10122* (2018).

[90] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *arXiv preprint arXiv:1801.01290* (2018).

[91] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination." In: *International Conference on Learning Representations*. 2019.

[92] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. "Learning latent dynamics for planning from pixels." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2555–2565.

[93] Tengda Han, Weidi Xie, and Andrew Zisserman. "Self-supervised co-training for video representation learning." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5679–5690.

[94] Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. "Fast task inference with variational intrinsic successor features." In: *arXiv preprint arXiv:1906.05030* (2019).

[95] Kyle Beltran Hatch, Sarthak J Shetty, Benjamin Eysenbach, Tianhe Yu, Rafael Rafailov, Ruslan Salakhutdinov, Sergey Levine, and Chelsea Finn. "Contrastive Example-Based Control." In: 2023.

[96] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.

[97] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. "Learning deep representations by mutual information estimation and maximization." In: *arXiv preprint arXiv:1808.06670* (2018).

[98] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning." In: *Advances in neural information processing systems* 29 (2016), pp. 4565–4573.

[99] Elad Hoffer and Nir Ailon. "Deep metric learning using triplet network." In: *International workshop on similarity-based pattern recognition*. Springer. 2015, pp. 84–92.

[100] Zhang-Wei Hong, Ge Yang, and Pulkit Agrawal. "Bilinear value networks." In: *arXiv preprint arXiv:2204.13695* (2022).

[101] Ferenc Huszár. "Variational inference using implicit distributions." In: *arXiv preprint arXiv:1702.08235* (2017).

[102] IQL Authors. Private Communication. 2022.

[103] Brian Ichter, Pierre Sermanet, and Corey Lynch. "Broadly-exploring, local-policy trees for long-horizon task planning." In: *arXiv preprint arXiv:2010.06491* (2020).

[104] Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. "On the convergence of stochastic iterative dynamic programming algorithms." In: *Neural computation* 6.6 (1994), pp. 1185–1201.

[105] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. "When to trust your model: Model-based policy optimization." In: *Advances in Neural Information Processing Systems*. 2019, pp. 12519–12530.

[106] Michael Janner, Igor Mordatch, and Sergey Levine. "gamma-models: Generative temporal difference learning for infinite-horizon prediction." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1724–1735.

[107] Joshua Joseph, Alborz Geramifard, John W Roberts, Jonathan P How, and Nicholas Roy. "Reinforcement learning with misspecified model classes." In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 939–946.

[108] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. "Exploring the limits of language modeling." In: *arXiv preprint arXiv:1602.02410* (2016).

[109] Leslie Pack Kaelbling. "Hierarchical learning in stochastic domains: Preliminary results." In: *Proceedings of the tenth international conference on machine learning*. Vol. 951. 1993, pp. 167–173.

[110] Leslie Pack Kaelbling. "Learning to achieve goals." In: *IJCAI*. Citeseer. 1993, pp. 1094–1099.

[111] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale." In: *ArXiv* abs/2104.08212 (2021).

[112] Hilbert J Kappen. "Path integrals and symmetry breaking for optimal control theory." In: *Journal of statistical mechanics: theory and experiment* 2005.11 (2005), P11011.

[113] Lydia Kavraki, Petr Svestka, and Mark H Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." In: *IEEE transactions on robotics and automation* 12.4 (1996), pp. 566–580.

[114] Michael Kearns and Satinder Singh. "Near-optimal reinforcement learning in polynomial time." In: *Machine learning* 49.2-3 (2002), pp. 209–232.

[115] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and T. Joachims. "MOReL : Model-Based Offline Reinforcement Learning." In: *ArXiv* abs/2005.05951 (2020).

[116] Leslie Kish. "Survey Sampling." In: *John Wiley & Sons* (1965).

[117] Mario Klingemann. *Raster Fairy*. https://github.com/bmcfee/RasterFairy. 2016.

[118] Vijay Konda and John Tsitsiklis. "Actor-critic algorithms." In: *Advances in neural information processing systems* 12 (1999).

[119] Ksenia Konyushkova, Konrad Zolna, Yusuf Aytar, Alexander Novikov, Scott Reed, Serkan Cabi, and Nando de Freitas. "Semi-supervised reward learning for offline reinforcement learning." In: *arXiv preprint arXiv:2012.06899* (2020).

[120] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. "Offline Reinforcement Learning with Implicit Q-Learning." In: *International Conference on Learning Representations*. 2021.

[121] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation." In: *Advances in neural information processing systems*. 2016, pp. 3675–3683.

[122] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. "Implicit under-parameterization inhibits data-efficient deep reinforcement learning." In: *arXiv preprint arXiv:2010.14498* (2020).

[123] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. "Model-ensemble trust-region policy optimization." In: *arXiv preprint arXiv:1802.10592* (2018).

[124] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. "Learning plannable representations with causal InfoGAN." In: *Advances in Neural Information Processing Systems*. 2018, pp. 8733–8744.

[125] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[126] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." In: *Advances in Neural Information Processing Systems*. 2017, pp. 6402–6413.

[127] Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. "Objective Mismatch in Model-based Reinforcement Learning." In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 761–770.

[128] Sascha Lange and Martin A Riedmiller. "Deep learning of visual control policies." In: *European Symposium on Artificial Neural Networks (ESANN)*. Citeseer. 2010. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.226.6898{\&}rep=rep1{\&}type=pdf.

[129] John Langford. *Specializations of the master problem*. 2010. URL: https://hunch.net/?p=1167.

[130] Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. "CIC: Contrastive Intrinsic Control for Unsupervised Skill Discovery." In: *Deep RL Workshop NeurIPS 2021*. 2021.

[131] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. "Curl: Contrastive unsupervised representations for reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5639–5650.

[132] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. "Reinforcement learning with augmented data." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19884–19895.

[133] Manfred Lau and James J Kuffner. "Behavior planning for character animation." In: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM. 2005, pp. 271–280.

[134] Yann LeCun. *Predictive learning*. https://www.youtube.com/watch?v=Ount2Y4qxQo. Keynote Talk. 2016.

[135] Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric Xing, and Ruslan Salakhutdinov. "Gated path planning networks." In: *arXiv preprint arXiv:1806.06408* (2018).

[136] Ian Lenz, Ross Knepper, and Ashutosh Saxena. "DeepMPC: Learning Deep Latent Features for Model Predictive Control." In: *Robotics: Science and Systems (RSS)*. 2015.

[137] Sergey Levine. "Reinforcement learning and control as probabilistic inference: Tutorial and review." In: *arXiv preprint arXiv:1805.00909* (2018).

[138] Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. "Space-time planning with parameterized locomotion controllers." In: *ACM Transactions on Graphics (TOG)* 30.3 (2011), p. 23.

[139] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. "Learning multi-level hierarchies with hindsight." In: *arXiv preprint arXiv:1712.00948* (2017).

[140] Andrew Levy, Robert Platt, and Kate Saenko. "Hierarchical Reinforcement Learning with Hindsight." In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ryzECoAcY7.

[141] Omer Levy and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." In: *Advances in neural information processing systems* 27 (2014).

[142] Alexander C Li, Lerrel Pinto, and Pieter Abbeel. "Generalized Hindsight for Reinforcement Learning." In: *arXiv preprint arXiv:2002.11708* (2020).

[143] Yitao Liang, Marlos C Machado, Erik Talvitie, and Michael Bowling. "State of the art control of Atari games using shallow reinforcement learning." In: *arXiv preprint arXiv:1512.01563* (2015).

[144] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." In: *ICLR (Poster)*. 2016.

[145] Xingyu Lin, Harjatin Singh Baweja, and David Held. "Reinforcement Learning without Ground-Truth State." In: *arXiv preprint arXiv:1905.07866* (2019).

[146] Xingyu Lin, Harjatin Singh Baweja, and David Held. "Reinforcement Learning without Ground-Truth State." In: *ArXiv* abs/1905.07866 (2019).

[147] Hao Liu and Pieter Abbeel. "Aps: Active pretraining with successor features." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6736–6747.

[148] Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. "Hallucinative topological memory for zero-shot visual planning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6259–6270.

[149] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. "Breaking the curse of horizon: Infinite-horizon off-policy estimation." In: *Advances in Neural Information Processing Systems*. 2018, pp. 5356–5366.

[150] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. "Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees." In: *ICLR (Poster)*. 2019.

[151] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. "Learning Latent Plans from Play." In: *arXiv preprint arXiv:1903.01973* (2019).

[152] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. "Learning latent plans from play." In: *Conference on Robot Learning*. 2020, pp. 1113–1132.

[153] Zhuang Ma and Michael Collins. "Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency." In: *EMNLP*. 2018.

[154] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. "Discovering and achieving goals via world models." In: *Advances in Neural Information Processing Systems* 34 (2021).

[155] Oliver Mihatsch and Ralph Neuneier. "Risk-sensitive reinforcement learning." In: *Machine learning* 49.2 (2002), pp. 267–290.

[156] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems* 26 (2013).

[157] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. "Learning to navigate in complex environments." In: *arXiv preprint arXiv:1611.03673* (2016).

[158] Andriy Mnih and Yee Whye Teh. "A fast and simple algorithm for training neural probabilistic language models." In: *ICML*. 2012.

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

[160] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. "Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections." In: *Advances in Neural Information Processing Systems*. 2019, pp. 2318–2328.

[161] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. "Data-Efficient Hierarchical Reinforcement Learning." In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 3303–3313.

[162] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. "Near-optimal representation learning for hierarchical reinforcement learning." In: *arXiv preprint arXiv:1810.01257* (2018).

[163] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7559–7566.

[164] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. "Visual Reinforcement Learning with Imagined Goals." In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 9191–9200.

[165] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Overcoming exploration in reinforcement learning with demonstrations." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6292–6299.

[166] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation." In: *Conference on Robot Learning*. PMLR. 2022, pp. 1303–1315.

[167] Suraj Nair, Silvio Savarese, and Chelsea Finn. "Goal-Aware Prediction: Learning to Model What Matters." In: *arXiv preprint arXiv:2007.07170* (2020).

[168] Soroush Nasiriany, Vitchyr H. Pong, Steven Lin, and Sergey Levine. "Planning with Goal-Conditioned Policies." In: *NeurIPS*. 2019.

[169] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. "Planning with goal-conditioned policies." In: *Advances in Neural Information Processing Systems*. 2019, pp. 14843–14854.

[170] Evgenii Nikishin, Romina Abachi, Rishabh Agarwal, and Pierre-Luc Bacon. "Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation." In: *arXiv preprint arXiv:2106.03273* (2021).

[171] Arnab Nilim and Laurent El Ghaoui. "Robustness in Markov Decision Problems with Uncertain Transition Matrices." In: *NIPS*. Citeseer. 2003, pp. 839–846.

[172] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. "f-GAN: Training generative neural samplers using variational divergence minimization." In: *Advances in neural information processing systems* 29 (2016).

[173] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. "Action-Conditional Video Prediction using Deep Networks in Atari Games." In: *Advances in Neural Information Processing Systems (NIPS)*. 2015. URL: https://arxiv.org/pdf/1507.08750v1.pdf.

[174] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. "Self-imitation learning." In: *arXiv preprint arXiv:1806.05635* (2018).

[175] Junhyuk Oh, Satinder Singh, and Honglak Lee. "Value prediction network." In: *arXiv preprint arXiv:1707.03497* (2017).

[176] Masashi Okada, Luca Rigazio, and Takenobu Aoshima. "Path integral networks: End-to-end differentiable optimal control." In: *arXiv preprint arXiv:1706.09597* (2017).

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

[178] Pedro A Ortega and Daniel A Braun. "Thermodynamics as a theory of decision-making with information-processing costs." In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469.2153 (2013), p. 20120683.

[179] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. "Deep exploration via bootstrapped DQN." In: *Advances in neural information processing systems*. 2016, pp. 4026–4034.

[180] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. "Time limits in reinforcement learning." In: *arXiv preprint arXiv:1712.00378* (2017).

[181] Ronald Parr and Stuart J Russell. "Reinforcement learning with hierarchies of machines." In: *Advances in neural information processing systems*. 1998, pp. 1043–1049.

[182] Keiran Paster, Sheila A McIlraith, and Jimmy Ba. "Planning from pixels using inverse dynamics models." In: *arXiv preprint arXiv:2012.02419* (2020).

[183] Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. "Maximum Entropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning." In: *arXiv preprint arXiv:2007.02832* (2020).

[184] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. "Multi-goal reinforcement learning: Challenging robotics environments and request for research." In: *arXiv preprint arXiv:1802.09464* (2018).

[185] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. "Skew-fit: State-covering self-supervised reinforcement learning." In: *arXiv preprint arXiv:1903.03698* (2019).

[186] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. "Temporal difference models: Model-free deep rl for model-based control." In: *arXiv preprint arXiv:1802.09081* (2018).

[187] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. "On variational bounds of mutual information." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5171–5180.

[188] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. "Data-efficient deep reinforcement learning for dexterous manipulation." In: *arXiv preprint arXiv:1704.03073* (2017).

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

[190] Shuang Qiu, Lingxiao Wang, Chenjia Bai, Zhuoran Yang, and Zhaoran Wang. "Contrastive UCB: Provably Efficient Contrastive Self-Supervised Learning in Online Reinforcement Learning." In: *International Conference on Machine Learning*. PMLR. 2022, pp. 18168–18210.

[191] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. "Imagination-augmented agents for deep reinforcement learning." In: *Advances in neural information processing systems*. 2017, pp. 5690–5701.

[192] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. "A game theoretic framework for model based reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7953–7963.

[193] Kate Rakelly, Abhishek Gupta, Carlos Florensa, and Sergey Levine. "Which Mutual-Information Representation Learning Objectives are Sufficient for Control?" In: *ArXiv* abs/2106.07278 (2021).

[194] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. "On stochastic optimal control and reinforcement learning by approximate inference." In: *Twenty-third international joint conference on artificial intelligence*. 2013.

[195] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. "Learning by playing solving sparse reward tasks from scratch." In: *International conference on machine learning*. PMLR. 2018, pp. 4344–4353.

[196] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[197] Stéphane Ross and Drew Bagnell. "Agnostic System Identification for Model-Based Reinforcement Learning." In: *ICML*. 2012.

[198] Tim GJ Rudner, Vitchyr H Pong, Rowan McAllister, Yarin Gal, and Sergey Levine. "Outcome-Driven Reinforcement Learning via Variational Inference." In: *arXiv preprint arXiv:2104.10190* (2021).

[199] Oleh Rybkin, Chuning Zhu, Anusha Nagabandi, Kostas Daniilidis, Igor Mordatch, and Sergey Levine. "Model-based reinforcement learning via latent-space collocation." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9190–9201.

[200] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. "Semi-parametric topological memory for navigation." In: *arXiv preprint arXiv:1803.00653* (2018).

[201] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. "Episodic Curiosity through Reachability." In: *arXiv preprint arXiv:1810.02274* (2018).

[202] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. "Universal value function approximators." In: *International conference on machine learning*. 2015, pp. 1312–1320.

[203] Karl Schmeckpeper, Annie Xie, Oleh Rybkin, Stephen Tian, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. "Learning predictive models from observation and interaction." In: *European Conference on Computer Vision*. Springer. 2020, pp. 708–725.

[204] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. "Mastering atari, go, chess and shogi by planning with a learned model." In: *Nature* 588.7839 (2020), pp. 604–609.

[205] Yannick Schroecker and Charles Isbell. "Universal value density estimation for imitation learning and goal-conditioned reinforcement learning." In: *arXiv preprint arXiv:2002.06473* (2020).

[206] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 815–823.

[207] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization." In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[208] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation." In: *arXiv preprint arXiv:1506.02438* (2015).

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

[210] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. "Time-contrastive networks: Self-supervised learning from video." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[211] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. "ViNG: Learning Open-World Navigation with Visual Goals." In: *arXiv preprint arXiv:2012.09812* (2020).

[212] Pararth Shah, Marek Fiser, Aleksandra Faust, J Chase Kew, and Dilek Hakkani-Tur. "Follownet: Robot navigation by following natural language directions with deep reinforcement learning." In: *arXiv preprint arXiv:1805.06150* (2018).

[213] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. "Dynamics-Aware Unsupervised Discovery of Skills." In: *International Conference on Learning Representations*. 2019.

[214] Rui Shu, Tung Nguyen, Yinlam Chow, Tuan Pham, Khoat Than, Mohammad Ghavamzadeh, Stefano Ermon, and Hung Bui. "Predictive coding for locally-linear control." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8862–8871.

[215] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search." In: *nature* 529.7587 (2016), p. 484.

[216] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. "Reward is enough." In: *Artificial Intelligence* 299 (2021), p. 103535.

[217] Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. "Identifying useful subgoals in reinforcement learning by local graph partitioning." In: *Proceedings of the 22nd international conference on Machine learning*. ACM. 2005, pp. 816–823.

[218] Kihyuk Sohn. "Improved Deep Metric Learning with Multi-class N-pair Loss Objective." In: *NeurIPS*. 2016.

[219] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. "Amortised map inference for image super-resolution." In: *arXiv preprint arXiv:1610.04490* (2016).

[220] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. "Semantic scene completion from a single depth image." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1746–1754.

[221] Jonathan Sorg, Satinder P Singh, and Richard L Lewis. "Internal rewards mitigate agent boundedness." In: *ICML*. 2010.

[222] A. Srinivas, A. Jabri, P. Abbeel, Sergey Levine, and Chelsea Finn. "Universal Planning Networks." In: *ArXiv* abs/1804.00645 (2018).

[223] Aravind Srinivas and Pieter Abbeel. *Unsupervised Learning for Reinforcement Learning*. Tutorial. 2021.

[224] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. "Universal planning networks: Learning generalizable representations for visuomotor control." In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4732–4741.

[225] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. "Training agents using upside-down reinforcement learning." In: *arXiv preprint arXiv:1912.02877* (2019).

[226] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. "Decoupling representation learning from reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9870–9879.

[227] Felipe Petroski Such, Vashisht Madhavan, Rosanne Liu, Rui Wang, Pablo Samuel Castro, Yulun Li, Jiale Zhi, Ludwig Schubert, Marc G Bellemare, Jeff Clune, et al. "An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents." In: *arXiv preprint arXiv:1812.07069* (2018).

[228] Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. "Policy Continuation with Hindsight Inverse Dynamics." In: *Advances in Neural Information Processing Systems*. 2019, pp. 10265–10275.

[229] Richard S Sutton. "Learning to predict by the methods of temporal differences." In: *Machine learning* 3.1 (1988), pp. 9–44.

[230] Richard S Sutton. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming." In: *Machine Learning Proceedings 1990*. Elsevier, 1990, pp. 216–224.

[231] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[232] Richard S Sutton and Brian Tanner. "Temporal-difference networks." In: *Advances in neural information processing systems*. 2005, pp. 1377–1384.

[233] Csaba Szepesvari, Richard S Sutton, Joseph Modayil, Shalabh Bhatnagar, et al. "Universal option models." In: *Advances in Neural Information Processing Systems*. 2014, pp. 990–998.

[234] Erik Talvitie. "Model Regularization for Stable Sample Rollouts." In: *UAI*. 2014, pp. 780–789.

[235] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. "Value iteration networks." In: *Advances in Neural Information Processing Systems*. 2016, pp. 2154–2162.

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

[237] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. "Distral: Robust multitask reinforcement learning." In: *Advances in Neural Information Processing Systems*. 2017, pp. 4496–4506.

[238] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. "A generalized path integral control approach to reinforcement learning." In: *The Journal of Machine Learning Research* 11 (2010), pp. 3137–3181.

[239] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive multiview coding." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer. 2020, pp. 776–794.

[240] Emanuel Todorov. "General duality between optimal control and estimation." In: *2008 47th IEEE Conference on Decision and Control*. IEEE. 2008, pp. 4286–4292.

[241] Y-H Tsai, H Zhao, M Yamada, L-P Morency, and R Salakhutdinov. "Neural Methods for Point-wise Dependency Estimation." In: *Proceedings of the Neural Information Processing Systems Conference (Neurips)*. 2020.

[242] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. "On mutual information maximization for representation learning." In: *arXiv preprint arXiv:1907.13625* (2019).

[243] Masatoshi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. "Generative adversarial nets from a density ratio estimation perspective." In: *arXiv preprint arXiv:1610.02920* (2016).

[244] Arun Venkatraman, Roberto Capobianco, Lerrel Pinto, Martial Hebert, Daniele Nardi, and J Andrew Bagnell. "Improved learning of dynamics models for control." In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 703–713.

[245] Srinivas Venkattaramanujam, Eric Crawford, Thang Van Doan, and Doina Precup. "Self-supervised Learning of Distance Functions for Goal-Conditioned Reinforcement Learning." In: *ArXiv* abs/1907.02998 (2019).

[246] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. "Feudal networks for hierarchical reinforcement learning." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3540–3549.

[247] Han Wang, Erfan Miahi, Martha White, Marlos C Machado, Zaheer Abbas, Raksha Kumaraswamy, Vincent Liu, and Adam White. "Investigating the Properties of Neural Network Representations in Reinforcement Learning." In: *arXiv preprint arXiv:2203.15955* (2022).

[248] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. "Unsupervised control through non-parametric discriminative rewards." In: *arXiv preprint arXiv:1811.11359* (2018).

[249] Christopher JCH Watkins and Peter Dayan. "Q-learning." In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[250] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images." In: *Advances in Neural Information Processing Systems (NIPS)*. 2015, pp. 2728–2736. arXiv: 1506.07365. URL: https://arxiv.org/pdf/1506.07365.pdfhttp://arxiv.org/abs/1506.07365.

[251] Kilian Q. Weinberger and Lawrence K. Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification." In: *NIPS*. 2005.

[252] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. "Information Theoretic MPC for Model-Based Reinforcement Learning." In: *International Conference on Robotics and Automation (ICRA)*. 2017.

[253] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[254] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. "Multi-task reinforcement learning: a hierarchical bayesian approach." In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 1015–1022.

[255] Yifan Wu, George Tucker, and Ofir Nachum. "The Laplacian in RL: Learning Representations with Efficient Approximations." In: *arXiv preprint arXiv:1810.04586* (2018).

[256] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. "Unsupervised Feature Learning via Non-parametric Instance Discrimination." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 3733–3742.

[257] Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn. "Few-shot goal inference for visuomotor learning and planning." In: *Conference on Robot Learning*. PMLR. 2018, pp. 40–52.

[258] Danfei Xu and Misha Denil. "Positive-unlabeled reward learning." In: *arXiv preprint arXiv:1911.00459* (2019).

[259] Ge Yang, Anurag Ajay, and Pulkit Agrawal. "Overcoming The Spectral Bias of Neural Value Approximation." In: *International Conference on Learning Representations*. 2021.

[260] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. "Mastering visual continuous control: Improved data-augmented reinforcement learning." In: *arXiv preprint arXiv:2107.09645* (2021).

[261] Denis Yarats, Ilya Kostrikov, and Rob Fergus. "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels." In: *International Conference on Learning Representations*. 2020.

[262] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images." In: *AAAI*. 2021.

[263] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. "Combo: Conservative offline model-based policy optimization." In: *arXiv preprint arXiv:2102.08363* (2021).

[264] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. "Gradient surgery for multi-task learning." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5824–5836.

[265] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning." In: *Conference on Robot Learning*. 2020, pp. 1094–1100.

[266] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. "MOPO: Model-based Offline Policy Optimization." In: *arXiv preprint arXiv:2005.13239* (2020).

[267] Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. "Composable planning with attributes." In: *arXiv preprint arXiv:1803.00512* (2018).

[268] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. "Learning Invariant Representations for Reinforcement Learning without Reconstruction." In: *International Conference on Learning Representations*. 2020.

[269] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. "SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning." In: *International Conference on Machine Learning (ICML)*. 2019. arXiv: 1808.09105. URL: http://arxiv.org/abs/1808.09105.

[270] Shangtong Zhang, Bo Liu, and Shimon Whiteson. "Gradientdice: Rethinking generalized offline estimation of stationary values." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11194–11203.

[271] Tianjun Zhang, Benjamin Eysenbach, Ruslan Salakhutdinov, Sergey Levine, and Joseph E Gonzalez. "C-Planning: An Automatic Curriculum for Learning Goal-Reaching Tasks." In: *International Conference on Learning Representations*.

[272] Tianjun Zhang, Tongzheng Ren, Mengjiao Yang, Joseph Gonzalez, Dale Schuurmans, and Bo Dai. "Making Linear MDPs Practical via Contrastive Representation Learning." In: *International Conference on Machine Learning*. PMLR. 2022, pp. 26447–26466.

[273] Rui Zhao, Xudong Sun, and Volker Tresp. "Maximum entropy-regularized multi-goal reinforcement learning." In: *arXiv preprint arXiv:1905.08786* (2019).

[274] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. "Target-driven visual navigation in indoor scenes using deep reinforcement learning." In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3357–3364.

[275] Brian D Ziebart. "Modeling purposeful adaptive behavior with the principle of maximum causal entropy." In: (2010).

[276] Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarej, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. "Task-relevant adversarial imitation learning." In: *arXiv preprint arXiv:1910.01077* (2019).