Bisimulation Makes Analogies in Goal-Conditioned Reinforcement Learning

Philippe Hansen-Estruch [1] **Amy Zhang** [1 2] **Ashvin Nair** [1] **Patrick Yin** [1] **Sergey Levine** [1]

Abstract

Building generalizable goal-conditioned agents from rich observations is a key to reinforcement learning (RL) solving real world problems. Traditionally in goal-conditioned RL, an agent is provided with the exact goal they intend to reach. However, it is often not realistic to know the configuration of the goal before performing a task. A more scalable framework would allow us to provide the agent with an example of an analogous task, and have the agent then infer what the goal should be for its current state. We propose a new form of state abstraction called *goal-conditioned bisimulation* that captures *functional equivariance*, allowing for the reuse of skills to achieve new goals. We learn this representation using a metric form of this abstraction, and show its ability to generalize to new goals in simulation manipulation tasks. Further, we prove that this learned representation is sufficient not only for goal-conditioned tasks, but is amenable to any downstream task described by a state-only reward function. Videos can be found at https://sites.google.com/view/gc-bisimulation.

1. Introduction

Goal-conditioned RL has the potential to train agents that can accomplish a variety of tasks when given a goal. However, the way the goal is represented in a goal-conditioned RL problem has a critical effect on how the resulting policy will interpret goals. For example, if a person is asked to accomplish the goal of cutting a vegetable, they can perform this task for any vegetable. The goal is represented in a way that is *invariant* to irrelevant factors, and potentially even *equivariant* to factors that vary between similar tasks, such as cutting long carrots versus short radishes. We use this equivariance to generalize to new problems and domains, by drawing associations to problems and domains we have experienced in the past. If we can acquire representations that have these properties, then we can similarly specify goals for goal-conditioned policies in one setting, and have them carry out those goals in many diverse settings. This frees us from needing an exact goal image for achieving the same functional task in a new setting. As shown in Figure 1, given an agent that has learned to cut carrots, rather than needing to present the agent with an image of chopped up radishes, we can instead say something along the lines of, "Do the same thing to the radish that you did to the carrot."

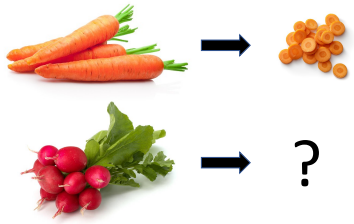


Figure 1. Analogous tasks of dicing carrots and radishes. Although the target objects are different, the skill required and functional difference between the initial state and goal images are similar. Our goal is to use this form of analogy to describe goals for new, unseen environments.

Prior work has proposed representation learning objectives that induce invariance to irrelevant factors through the use of domain knowledge about the relevant state features for a task (Jonschkowski and Brock, 2015; Jonschkowski et al., 2017), via data augmentations and contrastive learning (Laskin et al., 2020a), dynamics (Watter et al., 2015; Gelada et al., 2019), or proxy computer vision tasks (e.g., image reconstruction or segmentation) (Lange and Riedmiller, 2010; Sax et al., 2019). In goal-conditioned problems however, which features are relevant or irrelevant depends not only on the current state but also the goal. While there may be uncontrollable aspects of an environment that can always be ignored, knowledge of the task and current state is necessary to determine what is relevant to complete the task. Rather than construct representations that ignore features that are never relevant, our goal is to construct functionally equivariant representations that only capture changes be-

[1]University of California, Berkeley [2]Meta AI Research. Correspondence to: Philippe Hansen-Estruch <hansenpmeche@berkeley.edu>, Amy Zhang <amyzhang@fb.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Figure 2. An example of abstractions with a compositional form.

tween state-goal pairs. These equivariant representations can then be applied to depict goals for new states as a form of analogy. A key question therefore is: how do we design a form of representation learning objective that can perform this type of analogy?

In this work, we adapt a strict form of state abstraction called bisimulation (Larsen and Skou, 1989; Ferns et al., 2004) to address the goal-conditioned RL problem setting from rich observations. Rather than grouping equivalent states as bisimulation does, we group equivalent tasks — allowing policies to leverage analogous tasks when attempting new ones. One can also define a policy-dependent distance metric form of our goal-conditioned bisimulation (GCB) and use it to define a paired-state embedding space. We can then construct an objective for learning a state representation that can compose this task abstraction with states to depict new goals, capable of "filling in the blank", as shown in Figure 2. Such a representation is *universal* — it not only can be used for any goal-conditioned task, but is sufficient to train optimal policies for any state-only reward function.

Our contributions consist of the following: 1) an objective that learns a functionally equivariant abstraction over goal-conditioned tasks, 2) a demonstration that this embedding space is capable of composing states with task abstractions to depict and solve for new goals, 3) a novel value bound that can be applied to any downstream task, and 4) an evaluation on manipulation environments that shows improved performance on goal-conditioned tasks compared to other self-supervised representation learning methods.

2. Related Work

Our approach learns general-purpose representations for goal-conditioned RL based on principles from state aggregation. Our approach is orthogonal and complementary to goal-conditioned methods: we are not proposing a new algorithm for goal-conditioned RL, but rather a representation learning approach that can be used with goal-conditioned methods (and can also be used for general RL problems). Therefore, in our experiments, we compare to other representation learning approaches and use the same goal-conditioned RL algorithm for all baselines. We review prior work on representation learning here and more extensive prior work on goal-conditioned RL in Appendix B.

Work in **Representation learning for RL** can be decomposed into two categories: 1) state abstractions, which typically make use of task reward, and 2) self-supervised representation learning. As the scope of this body of work is very large, we focus on works exemplifying specific trends rather than a comprehensive review. Li et al. (2006) defined various forms of **state abstractions** for Markov decision processes (MDPs) that group states into clusters while preserving some property (e.g. the optimal value, all values, or all action values from each state). The strictest form, which generally preserves the most properties, is *bisimulation* (Larsen and Skou, 1989). Bisimulation only groups states that are indistinguishable w.r.t. reward sequences output given any action sequence tested. A related concept is bisimulation metrics (Ferns and Precup, 2014; Ferns et al., 2011), which measure how "behaviorally similar" states are. Gelada et al. (2019); Zhang et al. (2021); Agarwal et al. (2021); Castro et al. (2021) use behavior-based state abstractions to achieve better performance on control tasks by dropping irrelevant information.

Self-supervised RL methods rely on autoencoders (Lange and Riedmiller, 2010; Lange et al., 2012; Yarats et al., 2021b), contrastive objectives (van den Oord et al., 2018; Laskin et al., 2020a), data augmentation (Yarats et al., 2021a), and mutual information maximization (Anand et al., 2019; Choi et al., 2021) to learn representations for downstream control. Some work focuses on representation learning specifically for the goal-conditioned setting — Ghosh et al. (2019) uses goal-conditioned policies to extract a representation where Euclidean distances between states correspond to expected differences between actions to reach them. Similarly, Yang et al. (2020); Tian et al. (2021) use functional distances to plan to reach goals. However, none of these methods are capable of performing the types of analogies described in Section 1.

The idea of using **analogies** to improve learning is also not new (Carbonell, 1983), and much work has studied how compositionality and analogies arise naturally in learned representations. Mikolov et al. (2013) showed that latent language representations arising from skip-gram models were composable and produced analogies: e.g., $\phi(\text{"capital"}) + \phi(\text{"Germany"}) = \phi(\text{"Berlin"})$. Such analogies allow us to understand the latent space better and also demonstrate the structure of the latent space, which suggests it might be useful for downstream tasks. Similar analogies are also possible in visual domains (Reed et al., 2015; Jayaraman and Grauman, 2015). In control, analogy representations enabled goal-directed grasping (Jang et al., 2018) and composing plans (Devin et al., 2019). Similar in spirit to this work, our method imposes a network architecture to learn analogies, but enforces it through goal-conditioned bisimulation, making it a more general method for RL.

Metrics have also been leveraged in RL for both MDPs and states for the goal of positive transfer (Carroll and Seppi, 2005; Fernández and Veloso, 2006; Konidaris et al., 2012). Song et al. (2016) propose a metric between MDPs using

the Kantorovich and Hausdorff metrics and use it for transferring value functions between source tasks and a target task. Gelada et al. (2019) present results that their learned representation bounds the bisimulation metric and Zhang et al. (2021); Agarwal et al. (2021); Castro et al. (2021) use on-policy versions of the bisimulation metric (Castro, 2020) to learn an embedding space. Similar to Zhang et al. (2021), we embed our metric in a representation space and show improved performance with downstream control, but with goal-conditioned tasks and the use of analogies rather than robustness to distractors.

3. Preliminaries

We assume that the environment follows a **goal-conditioned Markov Decision Process** (GCMDP), which can be described with the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{G}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the dynamics model such that $\mathcal{P}(s'|s,a)$ dictates the probability of transitioning to state s' from state s after applying action a, $\mathcal{G} \subset \mathcal{S}$ is the goal space which is a subset of the state space, and $\mathcal{R}(s,a,g) := \mathbb{1}(s' = g)$ is a sparse reward function which returns 1 if (s,a) transitions to goal g and 0 otherwise. We refer to a state-goal pair (s,g) as a *task*, drawing connections to the single task RL setting where tasks are defined by state-action reward functions. While our primary concern is learning from images, we do not address the partial-observability problem explicitly: we follow prior work and use stacked pixel observations as the fully-observed system state s.

As noted in Section 2, **bisimulation** is a form of state abstraction that groups states $\mathbf{s}_i$ and $\mathbf{s}_j$ that are "behaviorally equivalent" (Li et al., 2006). For any action sequence $\mathbf{a}_{0:\infty}$, the probabilistic sequence of rewards from $\mathbf{s}_i$ and $\mathbf{s}_j$ are identical. A more compact definition has a recursive form: two states are bisimilar if they share both the same immediate reward and equivalent distributions over the next bisimilar states (Larsen and Skou, 1989; Givan et al., 2003).

Definition 3.1 (Bisimulation Relations (Givan et al., 2003))**.** Given an MDP $\mathcal{M}$, an equivalence relation B between states is a bisimulation relation if, for all states $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}$ that are equivalent under B (denoted $\mathbf{s}_i \equiv_B \mathbf{s}_j$) the following conditions hold:

$$\mathcal{R}(\mathbf{s}_i, \mathbf{a}) = \mathcal{R}(\mathbf{s}_j, \mathbf{a}) \quad \forall \mathbf{a} \in \mathcal{A}, \tag{1}$$

$$\mathcal{P}(G|\mathbf{s}_i, \mathbf{a}) = \mathcal{P}(G|\mathbf{s}_j, \mathbf{a}) \quad \forall \mathbf{a} \in \mathcal{A}, \quad \forall G \in \mathcal{S}_B, \tag{2}$$

where $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups G of equivalent states), and $\mathcal{P}(G|\mathbf{s},\mathbf{a}) = \sum_{\mathbf{s}' \in G} \mathcal{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})$.

Bisimulation relates functionally equivalent states together in a way can be usefully applied to representation learning. Exact partitioning with bisimulation relations is generally impractical in continuous state spaces, as the relation is highly sensitive to infinitesimal changes in the reward function or dynamics. For this reason, **Bisimulation Metrics** (Ferns et al., 2011; Ferns and Precup, 2014; Castro, 2020) softens the concept of state partitions, and instead defines a pseudometric space $(\mathcal{S}, d)$, where a distance function $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ measures the "behavioral similarity" between two states[1].

Computing bisimulation metrics is a harder class of problem than just finding an optimal policy due to the requirement that Equation (1) and Equation (2) hold true for $\forall \mathbf{a} \in \mathcal{A}$. In control problems, we typically only care about the optimal action $\mathbf{a}^*$ for any state. Castro (2020) define on-policy bisimulation, which we adopt to get the following on-policy metric:

$$d(\mathbf{s}_i, \mathbf{s}_j) := \mathbb{E}_{\mathbf{a} \sim \pi} |\mathcal{R}(\mathbf{s}_i, \pi^*(\mathbf{s}_i)) - \mathcal{R}(\mathbf{s}_j, \pi^*(\mathbf{s}_j))| + \gamma \cdot W_1(\mathcal{P}(\mathbf{s}_i, \pi^*(\mathbf{s}_i)), \mathcal{P}(\mathbf{s}_j, \pi^*(\mathbf{s}_j)); d). \tag{3}$$

However, this definition relies on a single reward function and focuses on grouping equivalent states. In Section 4.1 we explore how to modify this metric for goal-conditioned tasks in order to group equivalent (or analogous) tasks.

4. Functional Equivariance in an Embedding

We now build on the concepts introduced in Section 3 to step through a goal-conditioned metric in Section 4.1. Naïve lifting of the bisimulation metric into goal-conditioned MDPs gives us a metric over a *paired state space*, since it depends on both the current state and goal. This construction prevents compositional generalization to novel state-goal pairs, as we will show in Section 7. To obtain the type of compositional behavior we desire, we must leverage this back into a single state embedding. One form this can take is through simple arithmetic operations, where adding the task abstraction embedding to a single state embedding yields the desired goal. We will address how to construct an objective that gives rise to this property in Section 4.2. First, we introduce the basic version of applying bisimulation to goal-conditioned problems.

4.1. Defining a Goal-Conditioned Metric

Naïvely, we can apply state abstractions and metrics designed for single task settings to goal-conditioned ones by redefining them over pairs of states — a state-goal pair that defines a task. We walk through that process in this section.

Bisimulation is defined only for a single reward function, and therefore cannot be trivially applied to goal-conditioned settings. Furthermore, the central idea behind bisimulation

[1] Note that d is a pseudometric, meaning the distance between two different states can be zero.

is that it is an *equivalence relation* that allows us to cluster states to treat them analogously. We propose to lift this idea to families of tasks by defining an equivalence relation *over tasks*, giving us a functionally equivariant abstraction. Now that we have defined our environment in Section 3, we can now define a form of goal-conditioned bisimulation that defines an abstraction over state-goal pairs using a goal-conditioned reward function:

Definition 4.1 (Goal-conditioned Bisimulation Relations)**.** Given an MDP $\mathcal{M}$, as described in section 3, we define an equivalence relation B between states. This relation is a goal-conditioned bisimulation relation if, for all state-goal pairs $(\mathbf{s}_i, \mathbf{g}_i), (\mathbf{s}_j, \mathbf{g}_j) \in \mathcal{S} \bigcup \mathcal{G}$ that are equivalent under B (denoted $\mathbf{s}_i \equiv_B \mathbf{s}_j$) the following conditions hold:

$$\begin{aligned} \mathcal{R}(\mathbf{s}_i, \mathbf{a}, \mathbf{g}_i) &= \mathcal{R}(\mathbf{s}_j, \mathbf{a}, \mathbf{g}_j), \quad \forall \mathbf{a} \in \mathcal{A}, \\ \mathcal{P}(G|\mathbf{s}_i, \mathbf{a}) &= \mathcal{P}(G|\mathbf{s}_j, \mathbf{a}) \quad \forall \mathbf{a} \in \mathcal{A}, \forall G \in \mathcal{S}_B, \end{aligned}$$

Similar to in Section 3, we can define an on-policy version of this relation, which gives rise to a paired-state metric:

$$\begin{aligned} &d_\pi(\mathbf{s}_i, \mathbf{g}_i; \mathbf{s}_j, \mathbf{g}_j) = \\ &\quad |\mathcal{R}(\mathbf{s}_i, \pi(\mathbf{s}_i, \mathbf{g}_i), \mathbf{g}_i) - \mathcal{R}(\mathbf{s}_j, \pi(\mathbf{s}_j, \mathbf{g}_j), \mathbf{g}_j)| \\ &\quad + \gamma \mathcal{W}_2(\mathcal{P}(\mathbf{s}'_i|\mathbf{s}_i, \pi(\mathbf{s}_i, \mathbf{g}_i), \mathcal{P}(\mathbf{s}'_j|\mathbf{s}_j, \pi(\mathbf{s}_j, \mathbf{g}_j)) \end{aligned} \tag{4}$$

where π is the goal-conditioned policy and $\mathcal{W}_2$ is the Wasserstein distance. This distance metric captures functional equivalence across tasks, but is only defined for state-goal pairs and therefore cannot handle single states. We address this issue in the next section by using this metric to define a new state abstraction capable of forming analogies.

4.2. Defining a State Abstraction to Form Analogies

To generalize to novel state-goal combinations and make analogies across tasks, we need a single state representation represented by a state encoder ψ that still maintains the properties present in the state-goal representation ϕ. We take inspiration from works that have shown the arithmetic properties of embeddings (Mikolov et al., 2013; Jang et al., 2018) to enforce the same type of intuitive structure in the state encoder:

$$\psi(\mathbf{g}_i) - \psi(\mathbf{s}_i) = \phi(\mathbf{s}_i, \mathbf{g}_i), \quad \forall \mathbf{s}_i \in \mathcal{S}, \forall \mathbf{g}_i \in \mathcal{G}. \tag{5}$$

From Definition 4.1 we know that if there exists $\mathbf{s}_j \in \mathcal{S}, \mathbf{g}_j \in \mathcal{G}$ such that $\phi(\mathbf{s}_i, \mathbf{g}_i) = \phi(\mathbf{s}_j, \mathbf{g}_j)$, then

$$\psi(\mathbf{g}_j) - \psi(\mathbf{s}_j) := \psi(\mathbf{g}_i) - \psi(\mathbf{s}_i),$$

forming an "analogy" of state-goal pairs $(\mathbf{s}_i, \mathbf{g}_i)$ and $(\mathbf{s}_j, \mathbf{g}_j)$. An example of a form of analogy the representation constructed by ψ should be capable of is shown in Figure 2. By learning a single state representation ψ, we can use knowledge about different state-goal pairs and their properties to generalize to new state-goal pairs at evaluation time using a simple form of arithmetic in latent space. This property frees us from the traditional goal-conditioned structure, which requires access to the target goal we want an agent to reach. *Instead, an agent can reason about goals it has never seen before through examples of analogous tasks.*

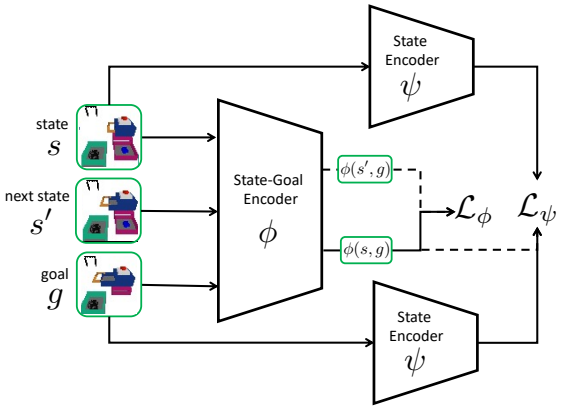


Figure 3. A flow diagram of the representation learning component of GCB. The dashed line represents stopped gradients and the state encoder ψ is a Siamese network using shared weights.

5. GCB: Constructing Embedding Spaces

We now present an algorithm that combines representation learning of the two components presented in Section 4 and downstream control for image-based goal-conditioned problems. In principle, GCB can be paired with any goal-conditioned RL method, including both online and offline algorithms. In the scope of our experiments, however, we focus on the offline setting to decouple from exploration difficulties. We first show how GCB utilizes the metric proposed in Section 4.1 to construct a single state representation amenable to generalization through the use of analogies.

5.1. Representation Learning Objectives

Our end goal is to construct a single state representation space with the properties introduced in Section 4.2. To do so, we must first learn an approximation of the policy-dependent GCB metric in Equation (4) end-to-end with our goal-conditioned policy π during offline RL training. We construct an embedding space ϕ where ℓ_1 distance between state-goal pairs corresponds to their distance under the GCB metric with the following objective:

$$\begin{aligned} \mathcal{L}_\phi = \Big(&||\phi(s_i, g_i) - \phi(s_j, g_j)||_1 - ||r_i - r_j||_2 \\ &- \gamma ||\bar{\phi}(s'_i, g_i) - \bar{\phi}(s'_j, g_j)||_2 \Big)^2, \end{aligned} \tag{6}$$

where $\bar{\cdot}$ denotes a stop-gradient. This objective is a direct instantiation of Equation (4) where, instead of learning a

Algorithm 1 GCB Training Algorithm

Given: RL algorithm $\mathcal{A}$, replay buffer $\mathcal{D} = \{(s, a, s', r, g)_i\}_{i=1}^N$, learning rates α_ψ, α_ϕ
Initialize π, ϕ, ψ
for training iteration $k = 1, 2, ...$ **do**
  $B_1 \leftarrow \{s_i, a_i, s'_i, r_i, g_i\}_{i=1}^B \sim \mathcal{D}\{\text{Sample Batch}\}$
  $B_2 \leftarrow \{s_j, a_j, s'_j, r_j, g_j\}_{j=1}^B = \text{permute } B_1$
  $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \mathcal{L}_\phi(B_1, B_2)$
  $\psi \leftarrow \psi - \alpha_\psi \nabla_\psi \mathcal{L}_\psi(B_1)$
  Update $\pi(\psi(s), \phi(s, g))$ with $\mathcal{A}$ on B_1
end for
for eval episode $m = 1, 2, ...$ **do**
  Sample analogous tasks (s, g) and (s_a, g_a)
  Rollout policy with $\pi(\psi(s), \phi(s_a, g_a))$
  Compute success of final achieved state against g
end for

parameterized distance function d_π, we instead directly optimize for an embedding space ϕ with the above property. We prove that this update procedure converges to our desired metric in Section 6.

We can now use this ϕ embedding to define an objective for constructing a single state representation that is capable of performing simple forms of "arithmetic" as described in Equation (5) in Section 4.2:

$$\mathcal{L}_\psi = \left((\bar{\phi}(s_i, g_i) - \bar{\phi}(g_i, g_i)) - (\psi(g_i) - \psi(s_i)) \right)^2. \quad (7)$$

Note that Equation (7) is a slightly different instantiation of Equation (5) where we add $\phi(g_i, g_i)$ as a normalizing constant. We found this to perform better empirically.[2] An ablation of this design decision can be found in Appendix D. This objective will imbue the single state representation ψ with the ability to form analogies across tasks.

In summary, GCB learns two representations: $\phi(s, g)$, which generalizes across state-goal pairs, and $\psi(s)$, which is shaped in latent space by ϕ but can generalize across single states. We will show in Section 7.2.1 that ϕ can capture *invariances* across tasks, which one can think of as only storing the "delta" between the initial state and goal, whereas ψ is capable of a form of arithmetic that captures *functional equivariance* across states and analogous tasks.

5.2. Combining GCB with Offline Downstream Control

In this section, we describe a way to combine GCB with goal-conditioned RL. Note that GCB is primarily a representation learning method, and is therefore agnostic to the downstream goal-conditioned RL algorithm. In the standard goal-conditioned setting we learn a policy $\pi(s, g)$, which with our learned representations becomes $\pi(\psi(s), \psi(g))$. We show in Section 7.2.2 that this enables better performance on goal-conditioned tasks.

However, GCB is designed primarily to solve tasks specified by analogies, where at test time the task goal g is unknown but instead specified by a separate state-goal pair (s_a, g_a) that achieves an analogous outcome with respect to *another* state s. Here, we wish to learn a policy that can condition on standard state-goal pairs $\pi(\psi(s), \phi(s, g))$ and successfully evaluate on an analogous pair $\pi(\psi(s), \phi(s_a, g_a))$. In Section 7.2.1, we see how GCB enables good performance when the policy is tasked by an analogous state-goal pair.

5.3. Architecture & Additional Implementation Details

As shown in Figure 3, GCB uses two encoders: $\phi(s, g)$, which acts as the paired state/goal encoder, and $\psi(s)$, which acts as the state encoder. Each is implemented as a convolutional neural network with six layers with 32 filters and map to latent spaces with dimensionality 256. We use Adam for optimization (Kingma and Ba, 2015). As we focus on the offline RL setting in our experiments, we implement our method on top of implicit Q-learning (IQL) (Kostrikov et al., 2021), a recent offline RL algorithm. The representation and policy are trained concurrently.

The ℓ_1 distance objective in Equation (6) is enforced with random pairs of transitions. As is standard in contrastive learning, to generate random pairs, we sample a batch $(s, a, s', r, g)_i$ from the replay buffer and randomly permute the samples in the batch as $(s, a, s', r, g)_j$.

In Appendix D we present ablations to evaluate several important implementation decisions: (1) Adding a grounding goal point $\phi(g_i, g_i)$ as a normalizing constant. (2) The choice of ℓ_1 distance versus ℓ_2 distance for matching the GCB metric. (3) Backpropagation of RL gradients to the encoder. (4) Adding a reward decoder model $\mathcal{R}(\phi(s, g), \phi(s', g))$ and (5) learned dynamics model as in DBC (Zhang et al., 2021). Additional implementation details and hyperparameters are in Appendix E.

Algorithm 1 details the training of GCB. The entire process is summarized as follows: 1) we sample a batch from the replay buffer, 2) randomly permute and match the batch and update ϕ according to Equation (6), 3) then update ψ using ϕ according to Equation (7), and finally 4) update π using the RL algorithm, where π can be parameterized either as $\pi(\psi(s_i), \psi(g_i))$ or $\pi(\psi(s_i), \phi(s_a, g_a))$ depending on the goal specification setting.

6. Value Bounds and Sufficiency Results

In this section, we first show that our update procedure for the goal-conditioned bisimulation metric has a unique fixed

[2] $\phi(g, g)$ is the universal goal point in the paired state encoder space, it should map to the same constant point for all goals.

point which is our desired metric. Further, we draw connections between value functions and this metric that allows us to provide transfer guarantees for a learned policy to new goals. These results are simple extensions of existing results on bisimulation metrics (Ferns et al., 2004). Finally, we also present a result on the universality of a representation learned with this metric that allows us to extend the above bounds to a larger class of downstream reward functions.

We first show that our goal-conditioned metric definition has a unique fixed point.

Proposition 6.1. *Let* $\mathfrak{met}$ *be the space of bounded pseudometrics on* $\mathcal{S}$ *and* π *a goal-conditioned policy that is continuously improving. Define* $\mathcal{F} : \mathfrak{met} \mapsto \mathfrak{met}$ *by*

$$\begin{aligned}\mathcal{F}(d,\pi)(\mathbf{s}_i,\mathbf{g}_i;\mathbf{s}_j,\mathbf{g}_j) = \\ (1-c)|\mathcal{R}(\mathbf{s}_i,\pi(\mathbf{s}_i,\mathbf{g}_i),\mathbf{g}_i) - \mathcal{R}(\mathbf{s}_j,\pi(\mathbf{s}_j,\mathbf{g}_j),\mathbf{g}_j)| \\ + cW(d)(\mathcal{P}^{\pi}_{\mathbf{s}_i},\mathcal{P}^{\pi}_{\mathbf{s}_j}),\end{aligned} \tag{8}$$

where $c \in (0,1)$ *is a metric discount factor. Then* $\mathcal{F}$ *has a least fixed point* d^{π} *which is a goal-conditioned* π*-bisimulation metric.*

Proof in Appendix C. This result shows that our update procedure in Equation (6) will converge to our desired metric. Next, we can show that, for any policy π, the GCB metric upper bounds the value difference between any two tasks, as described by state-goal pairs.

Proposition 6.2. *For any two state goal pairs* $(\mathbf{s}_i,\mathbf{g}_i),(\mathbf{s}_j,\mathbf{g}_j) \in (\mathcal{S},\mathcal{G})$ *and a given policy* π*,*

$$|V^{\pi}(\mathbf{s}_i,\mathbf{g}_i) - V^{\pi}(\mathbf{s}_j,\mathbf{g}_j)| \leq d^{\pi}(\mathbf{s}_i,\mathbf{g}_i;\mathbf{s}_j,\mathbf{g}_j). \tag{9}$$

Proof in Appendix C. For our learned policy π, this bounds the performance difference of it being applied to a new task denoted by state-goal pair (s_j,g_j), given its performance on a known task (s_i,g_i). Another way to interpret Proposition 6.2 is as an intuitive result similar to Bellman's (Bellman, 1957) policy improvement theorem and the extension in Barreto et al. (2017). *If two tasks have a small distance in* d^{π}*, the policy* π *will exhibit similar performance when deployed on them.*

We now show a novel result that the representations learned by ψ and ϕ are sufficient for any downstream task, not just goal-reaching tasks.

Proposition 6.3. *Assume that each state-action pair is visited infinitely often and the step-size parameters decay appropriately. Also assume that the goal space is the same as the state space, such that* $\mathcal{G} := \mathcal{S}$*. Learning with abstraction* ψ *for any reward function that can be expressed as* $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ *converges to the optimal state-action value function in the original MDP.*

Intuitively, Proposition 6.3 tells us that if our goal space is sufficiently rich, i.e., our agent learns to access every possible state in the MDP, then the information in ψ and ϕ is sufficient to train an optimal policy for any task that can be described by a state-only reward function.

Furthermore, we can extend these results to any downstream reward function. To do so, we first note that:

Remark 6.4. For a discrete state space $\mathcal{S}$ with cardinality $|\mathcal{S}|$, any reward function $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ can be written as a tabular combination of states in the form $\mathcal{R} = \{\alpha_i s_i\}_{i=0}^{|\mathcal{S}|}$, where $\alpha_i \in \mathbb{R}, \forall i$, and therefore there also exists a representation of that state space (such as one-hots) for which the reward can be written as a linear combination of states.

Then, we can also bound the value difference between two states $\mathbf{s}_i,\mathbf{s}_j \in \mathcal{S}$ for a given policy π and for any reward function as follows:

Proposition 6.5. *For any reward function* $R \in \mathcal{R}$ *as defined above and a given policy* π*,*

$$|V^{\pi}_R(\mathbf{s}_i) - V^{\pi}_R(\mathbf{s}_j)| \leq \sum_{k=0}^{|\mathcal{S}|} \alpha_k d^{\pi}(\mathbf{s}_i,\mathbf{g}_k;\mathbf{s}_j,\mathbf{g}_k), \tag{10}$$

where V^{π}_R *denotes the value function for a given policy* π *and reward function* R*.*

Thus, we have shown that not only is our representation learning objective sufficient for any downstream task, but is structured in a way that is amenable for *any* state-only reward function. One can view our method as a general, reward-agnostic representation learning scheme that exhibits nice properties for any downstream task. We now show the empirical capabilities of our method in Section 7.

7. Experiments and Results

We evaluate and compare GCB to other representation learning methods in the goal-conditioned setting. A central premise behind the design of GCB is that it can capture functional equivariance over different tasks, as described by state-goal pairs. Our experiments are therefore designed to answer the following questions: 1) Does GCB truly capture these types of analogies? 2) Can GCB use demonstration state-goal pairs to infer goals for novel states and successfully complete those tasks? 3) Can GCB also achieve strong performance on standard, offline goal-conditioned tasks?

7.1. Baselines and Prior Methods

We compare against representation learning methods following the major trends described in Section 2. We evaluate two contrastive approaches: a temporal contrastive method (CPC) (van den Oord et al., 2018) and a data augmentation contrastive method (CURL) (Laskin et al., 2020a). We also compare to reconstruction-based methods that use a VAE and context-conditioned VAE (CC-VAE (Nair et al., 2019)). Finally, we evaluate a pure data augmentation

method, RAD (Laskin et al., 2020b). All of the above representation learning methods produce single state representations, which we compare to our ψ representation directly. As an additional ablation, we can also evaluate the paired state representation ϕ we learn as an intermediate objective, which corresponds to a naïve goal-conditioned form of DBC (Zhang et al., 2021) which we denote by GC-DBC.

We also include another baseline that leverage compositionality by arithmetic sums over vectors, compositional plan vectors (CPV, (Devin et al., 2019)). However, CPVs are evaluated in a one-shot imitation learning setting which assumes access to paired reference trajectories. CPV uses an architecture $\pi(s_t, v_t)$ where $v_t = g(o_0, o_T) - g(o_0, o_t)$, where (o_0, o_t, o_T) are drawn from the reference trajectory in one-shot imitation learning to train $\pi(a_t|o_t)$. Our method does not have access to paired reference trajectories. Instead, we train the CPV architecture $\pi(s_t, v_t)$ with (o_0, o_t, o_T) being drawn from the current trajectory. This allows us to learn a compositional latent space for RL.

7.2. Simulation Manipulation Experiments

We evaluate GCB on complex manipulation tasks in a PyBullet simulated environment (Coumans and Bai, 2016–2021) consisting of randomly generated workspaces, as shown in Figure 4. We focus on the offline RL setting to decouple the exploration difficulties of this environment from the learning difficulties on various representations. Each initial state in this environment contains a randomized variety of objects and a randomly positioned drawer assembly, providing a variety of possible goals and distractors to evaluate functional equivariance and generalization.

Figure 4. `Button and Drawer` env with video distractors.

In the `Drawer` environment, a robot must learn to position a drawer in a target goal location, while in the other `Button and Drawer` environment, which consists of two stacked drawers with a button on top, the task of the robot is either to position the top drawer correctly or press the button which opens or closes a bottom drawer compartment. More details of the environment are in Appendix F. We create a distribution over tasks by randomizing the position, color, and orientation of the drawers. Additionally, we include between 0 and 4 distractor objects from a set of 84 object geometries, and in some experiments we also overlay distracting real video in the background (Zhang et al., 2018) from the Kinetics dataset (Kay et al., 2017). To collect the offline replay buffer, we use a noisy expert policy $\pi_{demo}(a_i|s, g) = \pi^*(a_i|s, g) + \mathcal{N}(0, 0.3)$ where $-1 \leq a_i \leq 1 \; \forall \; i$. 50K transitions are collected for the training set for the following offline RL experiments, where

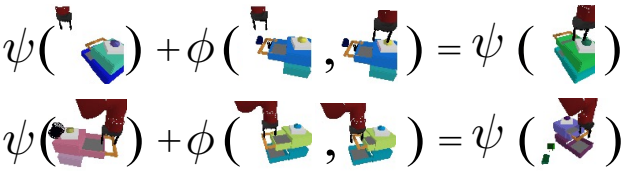


Figure 5. Two examples of **analogy arithmetic.** The rightmost image is the nearest neighbor in a test set of the composed representation in ψ space. Top: RHS has the finger pushing the button, just like in the goal argument to ϕ, but in a different position and orientation, indicating that the representation is equivariant to pose but captures the functional relevance of pushing the button. Bottom: RHS has the drawer being closed, again matching function with the goal argument to ϕ while exhibiting invariance to color and equivariance to pose.

the demonstrated policy reaches the goal in roughly 80% of attempted episodes. One epoch defines one full pass through the dataset. These environments were designed to contain several types of variation to showcase the different types of generalization GCB is capable of.

7.2.1. GCB Makes Analogies

In this section, we probe our learned representations ϕ and ψ to determine if they have the properties they were designed to have. In Figure 6 we sample random tasks, encode them with ϕ, and sample their nearest neighbors from a small subset of our offline dataset. According to Definition 4.1, these tasks being near each other in ϕ space means they are functionally similar. In comparing the sampled state-goal pairs, we see that GCB successfully captures *invariance*: features that are irrelevant to the task change across the sample and nearest neighbor in ϕ space, such as the color of the drawer and what distractor objects are present in the background. We show a similar result in Figure 7 but with the space of ϕ shown in a t-SNE plot. We again see that randomly sampled points in this space exhibit similar semantic tasks. For Figure 5, Figure 6, Figure 7, and Figure 8 the images shown are the ones given directly to the encoders ψ and ϕ. This visualization shows how the ϕ representation groups state-goal pairs and gives justification that GCB can make meaningful analogies. In our next visualization, we will show that ψ can exhibit *functional equivariance.*

While Figure 6 and Figure 7 show that our ϕ learning objective successfully groups equivalent tasks, we must now show that ψ is capable of *functional equivariance*, where pertinent factors of variation carry over across tasks. To test this, we compose example state-goal pairs with new states to describe new goals, in the manner shown in Figure 2. In Figure 5 we sample random states $s \in \mathcal{S}$ and an analogous state/goal (s_a, g_a) pair that is functionally similar but has a potentially different color drawer, background distractors, orientation, and/or position. We encode the state and analo-

Method	Drawer	Drawer+VD	BD	BD+VD	Analogy	Analogy+VD
GCB (ours)	$\mathbf{0.533 \pm 0.037}$	$\mathbf{0.448 \pm 0.033}$	0.440 ± 0.026	$\mathbf{0.322 \pm 0.021}$	$\mathbf{0.403 \pm 0.041}$	$\mathbf{0.303 \pm 0.027}$
CPV	0.400 ± 0.031	0.209 ± 0.013	0.228 ± 0.024	0.198 ± 0.013	0.176 ± 0.011	0.121 ± 0.009
GC-DBC	0.269 ± 0.025	0.261 ± 0.031	0.358 ± 0.023	0.255 ± 0.034	NA	NA
CCVAE	0.443 ± 0.029	0.231 ± 0.017	0.452 ± 0.039	0.225 ± 0.028	0.234 ± 0.012	0.095 ± 0.009
VAE	0.463 ± 0.023	0.261 ± 0.018	$\mathbf{0.522 \pm 0.042}$	0.115 ± 0.019	0.280 ± 0.024	0.112 ± 0.006
CPC	0.305 ± 0.020	0.207 ± 0.014	0.363 ± 0.042	0.154 ± 0.023	0.148 ± 0.026	0.145 ± 0.029
CURL	0.259 ± 0.021	0.283 ± 0.030	0.382 ± 0.038	0.198 ± 0.022	0.162 ± 0.021	0.123 ± 0.025
RAD	0.245 ± 0.019	0.207 ± 0.010	0.159 ± 0.036	0.108 ± 0.018	0.157 ± 0.028	0.110 ± 0.026
Pixel	0.483 ± 0.051	0.256 ± 0.027	0.117 ± 0.016	0.136 ± 0.027	0.132 ± 0.025	0.121 ± 0.033

Table 1. Final avg. success rates with stderr over 5 seeds. BD refers to the Button and Drawer env. and VD refers to Video Distractors.

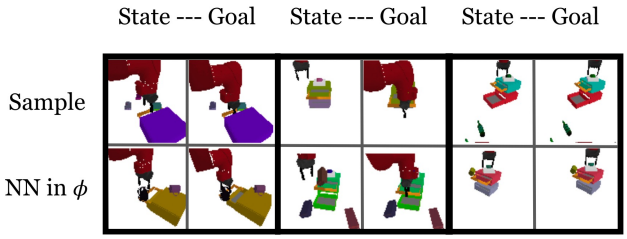


Figure 6. Visualization of nearest neighbors in ϕ. We sample state-goal pairs and find their NN in ϕ space. Features irrelevant to the task have changed, such as drawer color and distractor objects, but task-specific components are invariant, such as the relative positioning of robot to target object and semantics of the task.

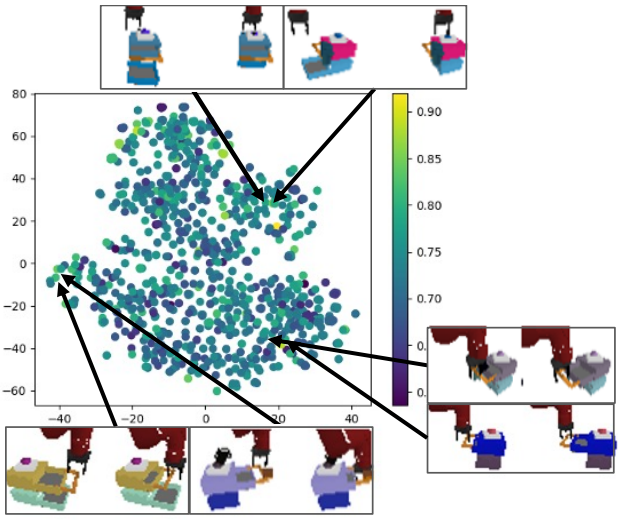


Figure 7. Visualization of nearest neighbors in ϕ as a t-SNE plot. Color corresponds to the predicted value output by the goal-conditioned critic.

gous state/goal, and add them to produce $\psi(s) + \phi(s_a, g_a)$. We then sample the nearest neighbor of this point in ψ space to find the closest representation of the desired goal. We find that ψ is able to take the analogous task and produce a reasonable goal for the current environment. If the analogous problem results in pressing the button, ψ then selects a similar state where the button is pressed (Figure 5, top), *maintaining the same orientation as the current initial state*. If it results in closing the drawer, the goal produced by ψ also corresponds to a state where the drawer is being closed (Figure 5, bottom). This indicates that the space is able to pick up on analogies, by representing tasks in an equivariant way such that applying the representation of a task from one initial state to another results in an analogous transformation to that state[3].

As discussed in Section 4.2, we can use analogies to describe goals for tasks where we do not have access to the true intended goal. Figure 5 constitutes a qualitative check that this is likely true, and we design the following experiment to quantitatively measure this ability. We collect an evaluation set where each initial state is paired with an analogous state-goal pair (s_a, g_a), where factors that the agent should be invariant or functionally equivariant to vary, similar to Figure 5. The policy is trained with $\pi(\psi(s), \phi(s, g))$, but is evaluated with $\pi(\psi(s), \phi(s_a, g_a))$. We focus on the Button and Drawer environment as it has a richer space of goals — the policy has to use information from s_a, g_a to infer the correct goal. Success in this environment requires representations that encode important task information across visually different tasks.

As quantitative evidence of GCB's ability to use analogies, we see in Table 1 that GCB remains the best performer, achieving 40% success on average. This shows that GCB can map analogous problems to the current environment, successfully reaching goals that it was never given goal images of — rather only an analogous task as an example. In the next section, we see if these properties also help GCB on standard goal-conditioned RL tasks.

7.2.2. Comparisons on goal-reaching

We have shown that GCB exhibits invariance and functional equivariance across tasks, which leads to superior perfor-

[3] We do not see the exact goal that matches the given initial state because it is not part of the dataset we sample from.

mance in a task setting where the agent is given an analogous task as a goal descriptor rather than the specified goal image at evaluation time. Now we ask, does GCB also lead to improved performance on standard goal-reaching tasks? We evaluate in the standard goal-conditioned setting on new, unseen environments where we provide the policy $\pi(s, g)$ the intended true goal image g.

In the standard goal setting without video distractors, GCB is able to outperform all other representation methods on the `Drawer` benchmark achieving 50%+ success, as seen in Table 1. In `Button and Drawer`, GCB maintains strong performance, beating all representations except for VAE and CCVAE, where it remains competitive. When real video distractors are added, GCB performs the best in both tasks. The performance of GC-DBC shows the necessity of the compositional single state representation, ψ — without it, we are not able to achieve as strong performance on new, unseen environments, as was noted in Section 4.

8. Discussion

This paper introduces goal-conditioned bisimulation (GCB), a representation learning method for goal-conditioned RL problems that captures functional equivariance over a family of goal-conditioned tasks. We have shown that, at evaluation time, an agent can use representations of analogous tasks to achieve unseen goals and even outperform existing representation learning methods in the standard goal-conditioned setting. Furthermore, our method is theoretically backed by an equivalence relation that defines what "functional similarity" means. The representations learned by our method are structured, in that ℓ_1 distance in the space bounds the value difference of different states and tasks. This holds true not only for goal-conditioned problems, but all downstream tasks describable by a state-only reward function.

While we show improved success rates in goal-conditioned tasks and improved generalization capability on unseen, in-distribution environments, it is possible that we can broaden the definition of functional equivalence beyond that of goal-conditioned bisimulation. As an example, one limitation of GCB is that the number of steps the policy needs to complete the task is a determining factor in how close two tasks are. While it is not entirely clear how we can broaden this equivalence relation while maintaining some semantic meaning of *function*, the potential of such an abstraction is plain. We can define equivalences across tasks in different domains and action spaces, allowing agents to learn from third-person demonstrations and unlocking the potential of large, off-domain datasets.

Acknowledgements

This research was supported by a Meta AI-BAIR Commons project, AN is supported by the Army Research Office under W911NF-21-1-009, and the Office of Naval Research, with compute support from Google Cloud, Berkeley Research Computing, Meta, and Azure.

References

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised State Representation Learning in Atari. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob Mcgrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Andre Barreto, Will Dabney, Remi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor Features for Transfer in Reinforcement Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

Jaime Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. 12 1983.

James Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *In Proceedings of the International Joint Conference on Neural Networks*, pages 803 – 808 vol. 2, 01 2005.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for markov decision

processes. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jacob Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1518–1528. PMLR, 2021.

Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. Variational empowerment as representation learning for goal-conditioned reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1953–1963. PMLR, 18–24 Jul 2021.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Coline Devin, Daniel Geng, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Compositional Plan Vectors. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Ben Eysenbach, Xinyang Geng, Sergey Levine, and Russ R Salakhutdinov. Rewriting History with Inverse RL: Hindsight Inference for Policy Improvement. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14783–14795. Curran Associates, Inc., 2020.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, page 720–727, New York, NY, USA, 2006. Association for Computing Machinery.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*, pages 162–169, 2004.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *Society for Industrial and Applied Mathematics*, 40(6): 1662–1714, December 2011.

Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Uncertainty in Artificial Intelligence (UAI)*, pages 210–219, 2014.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning (ICML)*, 2019.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal conditioned policies. In *International Conference on Learning Representations*, 2019.

Robert Givan, Thomas L. Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147:163–223, 2003.

Beining Han, Chongyi Zheng, Harris Chan, Keiran Paster, Michael R. Zhang, and Jimmy Ba. Learning domain invariant representations in goal-conditioned block MDPs. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. Grasp2vec: Learning object representations from self-supervised grasping. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 99–112. PMLR, 2018.

Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to egomotion. In *ICCV*, 2015.

Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39: 407–428, 10 2015.

Rico Jonschkowski, Roland Hafner, Jonathan Scholz, and Martin Riedmiller. Pves: Position-velocity encoders for unsupervised learning of structured state representations. 5 2017.

L P Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume vol.2, pages 1094 – 8, 1993.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *Computing Research Repository (CoRR)*, 2017.

Alexander Khazatsky, Ashvin Nair, Daniel Jing, and Sergey Levine. What can I do here? learning new skills by

imagining visual affordances. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 14291–14297. IEEE, 2021.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

George Konidaris, Ilya Scheidwasser, and Andrew G. Barto. Transfer in reinforcement learning via shared features. *J. Mach. Learn. Res.*, 13:1333–1371, May 2012.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021.

Sascha Lange and Martin A. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *IJCNN*, pages 1–8. IEEE, 2010.

Sascha Lange, Martin Riedmiller, Arne Voigtlander, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks (IJCNN)*, number June, pages 1–8. IEEE, 2012.

K. G. Larsen and A. Skou. Bisimulation through probabilistic testing (preliminary report). In *Symposium on Principles of Programming Languages*, page 344–352. Association for Computing Machinery, 1989.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 5639–5650. PMLR, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement Learning with Augmented Data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020b.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.

Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for zero-shot visual planning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6259–6270. PMLR, 13–18 Jul 2020.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Ofir Nachum, Shane Shixiang Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ashvin 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 (NeurIPS)*, 2018.

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. In *Conference on Robot Learning (CoRL)*, 2019.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with Goal-Conditioned Policies. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Scott E. Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. *Advances in Neural Information Processing Systems*, 28, 2015.

Alexander Sax, Bradley Emi, Amir Zamir, Leonidas Guibas, Silvio Savarese, and Jitendra Malik. Mid-Level Visual Representations Improve Generalization and Sample Efficiency for Learning Visuomotor Policies. In *Conference on Robot Learning (CoRL)*, 2019.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*, pages 1312–1320, 2015.

Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *5th Annual Conference on Robot Learning*, 2021.

Jinhua Song, Yang Gao, Hao Wang, and Bo An. Measuring the distance between finite markov decision processes. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '16, page 468–476, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems.

Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2021.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. 2018.

Angelina Wang, Thanard Kurutach, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

David Warde-Farley, Tom Van De Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Mnih Volodymyr. Unsupervised Control Through Non-Parametric Discriminative Rewards. In *International Conference on Learning Representations (ICLR)*, 2019.

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 (NeurIPS)*, pages 2728–2736, 2015.

Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2vec: Unsupervised representation learning by latent plans. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 935–946. PMLR, 10–11 Jun 2020.

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*, 2021a.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021b.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *CoRR*, abs/1811.06032, 2018.

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*, 2021.

A. Extra Visualizations

We add extra Nearest Neighbors visualizations for our arithmetic space. These follow the same logic as shown in Figure 5. The analogies are sampled by randomizing color, distractor objects, video distractors (if present), and translating and rotating the drawer. We provide more details about how we sample these analogies in Section 7.2.1.

Figure 8. More examples of **analogy arithmetic.** The leftmost image is a sampled start state. The next two images are a sampled analogy start and goal. The rightmost image is the nearest neighbor in a test set of the composed representation in ψ space. Each row is a new sample, the second to last row is an example of a failure.

B. Extended Related Work

Goal-Conditioned RL. Goal-conditioned RL learns a policy that can generalize to many reward functions, indexed by goals, rather than a single reward function (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017; Nachum et al., 2018; Eysenbach et al., 2020). For environments with rich, high-dimensional observations such as images, methods

have proposed to use existing generative models for encoding observations and goals (Nair et al., 2018; 2019; Khazatsky et al., 2021). Using reconstructive generative models for the input representation compresses the state space by encoding appearance information, but does not relate to functionality. Goal-conditioned RL has been used to learn general policies in the real world on large, diverse datasets (Chebotar et al., 2021). Thus making goal-conditioned RL more sample efficient and performant is a practical contribution towards real-world reinforcement learning. Non-parametric methods for exploration and planning with images have been considered (Warde-Farley et al., 2019; Nasiriany et al., 2019; Wang et al., 2019; Liu et al., 2020; Shah et al., 2021) and could also benefit from improved representation learning. Han et al. (2021) address the generalization problem of goal-conditioned policies to novel observations at test time by using an alignment objective for data collection across environments. Our method instead naturally finds alignment through an equivalence relation.

C. Proofs of Theoretical Results

Theorem C.1. *Define $\mathcal{F}^\pi : \mathcal{M} \to \mathcal{M}$ by $\mathcal{F}^\pi(d)(s,t) = |\mathcal{R}^\pi_s - \mathcal{R}^\pi_t| + \gamma \mathcal{W}_1(d)(\mathcal{P}^\pi_s, \mathcal{P}^\pi_t)$, then $\mathcal{F}^\pi$ has a least fixed point d^π, and d^π is a π-bisimulation metric.*

Proposition 6.1.

$$\mathcal{F}(d,\pi)(\mathbf{s}_i, \mathbf{g}_i; \mathbf{s}_j, \mathbf{g}_j) = (1-c)|\mathcal{R}(\mathbf{s}_i, \pi(\mathbf{s}_i, \mathbf{g}_i), \mathbf{g}_i) - \mathcal{R}(\mathbf{s}_j, \pi(\mathbf{s}_j, \mathbf{g}_j), \mathbf{g}_j)| + cW(d)(\mathcal{P}^\pi_{\mathbf{s}_i}, \mathcal{P}^\pi_{\mathbf{s}_j}).$$

Then $\mathcal{F}$ has a least fixed point d^π which is a goal-conditioned π-bisimulation metric.

Proof. We start from the result in Theorem 2 of Castro (2020), also included as Theorem C.1 here. We can reconstruct our GCMDP as an equivalent, standard super-MDP $\mathcal{M}'$ by concatenating the state and goal spaces to construct a new state space: $\mathcal{S}' := \mathcal{S} \times \mathcal{G}$ with new reward function $\mathcal{R}' := \mathbb{1}(s' = (g, g))$. We can then rewrite $\mathcal{F}$ as:

$$\mathcal{F}_{\mathcal{M}'}(d,\pi)(\mathbf{s}'_i, \mathbf{s}'_j) = (1-c)|\mathcal{R}'(\mathbf{s}'_i, \pi(\mathbf{s}'_i) - \mathcal{R}'(\mathbf{s}'_j, \pi(\mathbf{s}'_j))| + cW(d)(\mathcal{P}^\pi_{\mathbf{s}'_i}, \mathcal{P}^\pi_{\mathbf{s}'_j}).$$

It should be clear that this is equivalent to the $\mathcal{F}$ defined for the GCMDP as the reward functions are identical and the dynamics do not change, since $\mathcal{P}$ does not depend on the goal. We can then apply Theorem C.1 to this super-MDP to show that $\mathcal{F}_{\mathcal{M}'}$ has a unique fixed point which is the π-bisimulation metric, proving that the equivalent $\mathcal{F}$ also has the same unique fixed point. □

Proposition C.2. *For any $\pi \in \mathscr{P}(\mathcal{A})^\mathcal{S}$ and states $x, y \in \mathcal{S}$, we have $|V^\pi(x) - V^\pi(y)| \leq d^\pi(x,y)$.*

Proposition 6.2. *For any two state goal pairs $(\mathbf{s}_i, \mathbf{g}_i), (\mathbf{s}_j, \mathbf{g}_j) \in (\mathcal{S}, \mathcal{G})$,*

$$|V^\pi(\mathbf{s}_i, \mathbf{g}_i) - V^\pi(\mathbf{s}_j, \mathbf{g}_j)| \leq d^\pi(\mathbf{s}_i, \mathbf{g}_i; \mathbf{s}_j, \mathbf{g}_j).$$

Proof. We start from the result in Proposition 4.8 in Castro et al. (2021), also reproduced here as Proposition C.2. We again reconstruct our GCMDP as an equivalent, standard super-MDP $\mathcal{M}'$ by concatenating the state and goal spaces to construct a new state space: $\mathcal{S}' := \mathcal{S} \times \mathcal{G}$ with new reward function $\mathcal{R}' := \mathbb{1}(s' = (g, g))$. We can now apply Proposition C.2 to this super-MDP, showing the above bound to be true. □

Proposition 6.3. *If we assume that the goal space is the same as the state space, i.e. $\mathcal{G} := \mathcal{S}$, then GCB learns a representation sufficient to learn the optimal policy for any downstream reward function that can be expressed as $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$.*

Proof. As noted in Remark 6.4, a state-only reward function can be written as a set of all states and their rewards, $\mathcal{R} = \{\alpha_i s_i\}_{i=0}^{|\mathcal{S}|}$, where $\alpha_i \in \mathbb{R}, \forall i$. The optimal policy π^* for any reward function $\mathcal{R}$ can then be expressed as a series of state-reaching policies, which will be within the set of goal-conditioned policies where $\mathcal{G} = \mathcal{S}$. □

Proposition 6.5. *For any reward function $R \in \mathcal{R}$ as defined above and a given policy π,*

$$|V^\pi_R(\mathbf{s}_i) - V^\pi_R(\mathbf{s}_j)| \leq \sum_{k=0}^{|\mathcal{S}|} \alpha_k d^\pi(\mathbf{s}_i, \mathbf{g}_k; \mathbf{s}_j, \mathbf{g}_k).$$

Proof. Without loss of generality, we assume $\alpha_i \geq 0, \forall i$. Starting from Proposition 6.2 we have that $\forall g \in \mathcal{S}$,

$$|V^\pi(\mathbf{s}_i, \mathbf{g}) - V^\pi(\mathbf{s}_j, \mathbf{g})| \leq d^\pi(\mathbf{s}_i, \mathbf{g}; \mathbf{s}_j, \mathbf{g}). \tag{11}$$

We can express $V_R^\pi(\mathbf{s}_i)$ as a sum over goal-conditioned values:

$$V_R^\pi(\mathbf{s}_i) = \sum_{k=0}^{|\mathcal{S}|} \alpha_k V^\pi(\mathbf{s}_i, \mathbf{g}_k).$$

We can plug this into the LHS:

$$|V_R^\pi(\mathbf{s}_i) - V_R^\pi(\mathbf{s}_j)| = \left| \sum_{k=0}^{|\mathcal{S}|} \alpha_k V^\pi(\mathbf{s}_i, \mathbf{g}_k) - \sum_{k=0}^{|\mathcal{S}|} \alpha_k V^\pi(\mathbf{s}_j, \mathbf{g}_k) \right|.$$

Combine terms into one summation,

$$|V_R^\pi(\mathbf{s}_i) - V_R^\pi(\mathbf{s}_j)| = \left| \sum_{k=0}^{|\mathcal{S}|} \alpha_k \big(V^\pi(\mathbf{s}_i, \mathbf{g}_k) - V^\pi(\mathbf{s}_j, \mathbf{g}_k) \big) \right|.$$

Then use the triangle inequality to move the absolute value inside the sum,

$$|V_R^\pi(\mathbf{s}_i) - V_R^\pi(\mathbf{s}_j)| \leq \sum_{k=0}^{|\mathcal{S}|} \alpha_k \big| V^\pi(\mathbf{s}_i, \mathbf{g}_k) - V^\pi(\mathbf{s}_j, \mathbf{g}_k) \big|.$$

Plugging in Equation (11),

$$|V_R^\pi(\mathbf{s}_i) - V_R^\pi(\mathbf{s}_j)| \leq \sum_{k=0}^{|\mathcal{S}|} \alpha_k d^\pi(\mathbf{s}_i, \mathbf{g}_k; \mathbf{s}_j, \mathbf{g}_k).$$

□

D. Additional Ablations

In this section, we run experiments that modify various aspects of our algorithm to show how different design choices and hyperparameters affect performance.

D.1. Dimensionality of ϕ and ψ

We start by examining how changing the latent dimensionality of the two spaces we learn, denoted by ϕ and ψ, affect downstream success rate on the `Drawer` task. In Figure 9 we find that this hyperparameter does not significantly affect downstream performance of the control task. All other ablations are performed on the `Button and Drawer`.

D.2. Grounding ψ with Goal Point $\phi(g, g)$

As discussed in Section 5, the loss for ψ in Equation (7) which differs from the one proposed in Equation (5). Considering ϕ is trained with pairs, $\phi(s, g)$ alone might not properly represent an informative point. Instead, $\phi(s, g) - \phi(g, g)$ will more likely have a well defined distance as it involves a comparison of pairs or tasks. We refer to this as grounding ψ with the goal point. We ablate on this difference in Figure 10 by running GCB where ψ's objective is either Equation (7) (GCB-G) or Equation (5) (GCB-NG). Grounding ψ has a strong performance boost compared to not grounding.

D.3. Using ℓ_1 vs ℓ_2 Norm Objective

ϕ's loss in Equation (6) uses an ℓ_1 metric loss on pairs to fit ϕ's encoder. In this section we ablate on using ℓ_1 or ℓ_2 ($||\phi(s_i, g_i) - \phi(s_j, g_j)||_2$)) for the first portion of ϕ's loss. We report results in Figure 11. There is a large difference between the two losses, ℓ_1 achieving far better performance. We suspect that ℓ_1 results in more stable training of ϕ.

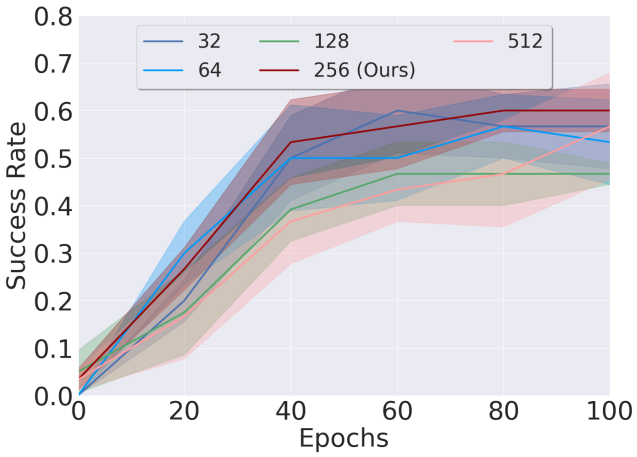


Figure 9. Ablation on different dimensionality for ϕ and ψ. We see that our tuned hyperparameter of 256 performs best, but the range of values from 32 to 512 generally does not affect success rate significantly.

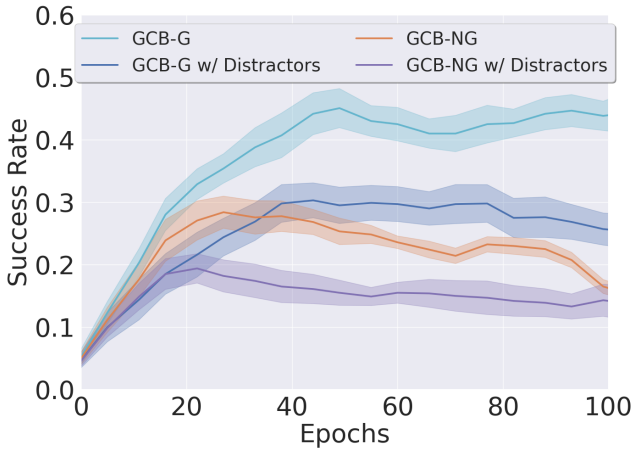


Figure 10. Ablation on different ψ objectives. GCB-G uses the grounded loss and GCB-NG does not. Video Distractors are added for some ablations.

D.4. Backpropagating RL Objectives through GCB

All other baselines compared to in Section 7 allow critic gradients from IQL's Q function into the encoder. We ablate using critic gradients with Equation (7) to see if it is helpful. Results are shown in Figure 12. Critic gradients seem marginally harmful for downstream performance, so we decided to not use them in GCB.

D.5. Adding a Reward Decoder

In the DBC implementation, a reward decoder is used to help stabilize the representation (Zhang et al., 2021). Considering that rewards are sparse in our domain, it might be harder to train a decoder. We ablate using and not using a reward decoder. The reward decoder $\mathcal{R}(\phi(s, g), \phi(s', g))$ is an MLP that is trained to predict if a task transition goes to the goal. The reward decoder loss is shown in Equation 12 and the gradient backpropagates through the ϕ encoder.

$$\mathcal{L}_{\mathcal{R}} = \left(\mathcal{R}(\phi(s_i, g_i), \phi(s'_i, g_i)) - r_i\right)^2, \tag{12}$$

We report results in Figure 13. The results show that GCB has very little performance difference with and without a reward

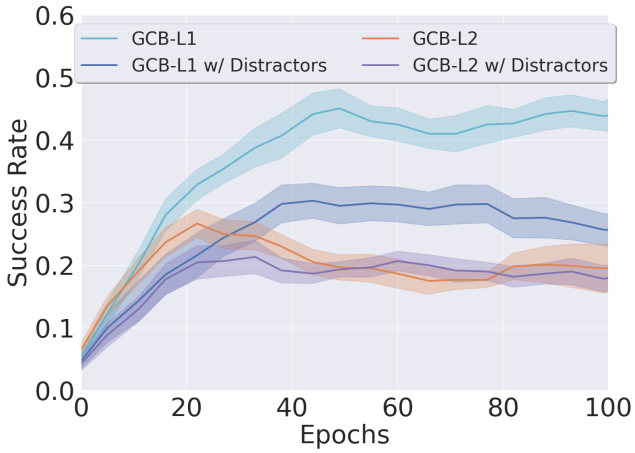


Figure 11. Ablation on different ϕ objectives. GCB-ℓ_1 uses an ℓ_1 metric loss and GCB-ℓ_2 uses ℓ_2 . Video Distractors are added for some ablations.

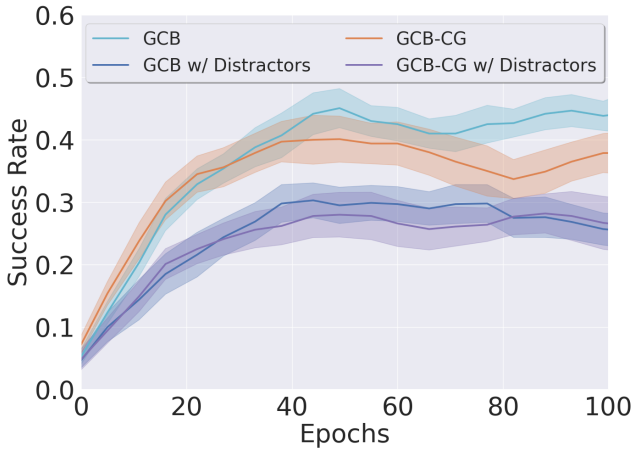


Figure 12. Ablation on different ϕ objectives. GCB uses no critic gradients and GCB-CG uses critic gradients. Video Distractors are added for some ablations.

decoder. We decided to leave it in the implementation used for the main results.

D.6. Using Dynamics Model vs Next Observation GCB

In the definitions of bisimulation metrics discussed in Section 3 and Section 4, a dynamics model $\mathcal{P}$ was used to define similar next states. There are two possible instantiations for implementing a learned version of this metric: learning a dynamics model that outputs a Gaussian next state distribution and calculating the Wasserstein, as proposed in Zhang et al. (2021), or using sampled interactions, as introduced in Castro et al. (2021). While we took the latter approach as our main method, we investigate using a learned dynamics approach here. An MLP dynamics model $f(\phi(s, g), a)$ could be learned and used in second portion of our loss in Equation 6. We define that new ϕ loss as follows:

$$\mathcal{L}_\phi = \Bigg(||\phi(s_i, g_i) - \phi(s_j, g_j)||_1 - ||r_i - r_j||_2 \\ - \gamma ||f(\bar{\phi}(s_i, g_i), a_i) - f(\bar{\phi}(s_j, g_j), a_j) + \bar{\phi}(s_i, g_i) - \bar{\phi}(s_j, g_j)||_2 \Bigg)^2, \tag{13}$$

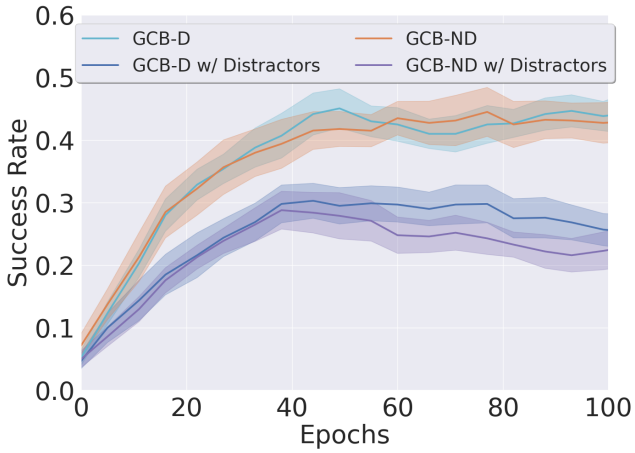


Figure 13. Ablation on different ϕ objectives. GCB-D uses a reward decoder and GCB-ND doesn't use a reward decoder. Video Distractors are added for some ablations.

as well as a dynamics loss for training the model f:

$$\mathcal{L}_f = \Big(f(\phi(s,g),a) + \phi(s',g) - \phi(s,g) \Big)^2.$$

We present ablations of using the standard Equation 6 loss versus using a dynamics model and 13 in Figure 14. We find that using next observation results in a more stable learning process compared to using a learned dynamics model, although the dynamics model method is capable of achieving similar peak performance.

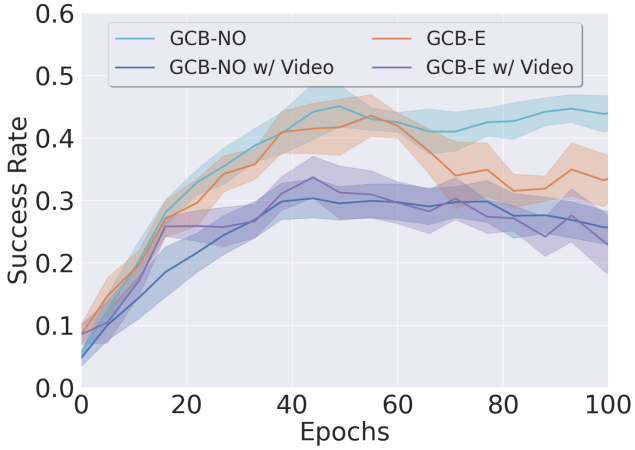


Figure 14. Ablation on different ϕ objectives. GCB-NO uses next observation and GCB-E uses the dynamics model based loss. Video Distractors are added for some ablations.

E. Implementation Details

We use a similar encoder architecture to (Laskin et al., 2020b) except we use a six layer convolutional network. The encoder has kernels of size 3×3 with 32 channels for all the convolutional layers and set stride to 1 everywhere, except of the first convolutional layer, which has stride 2, and interpolate with ReLU activations. We modify a PyTorch implementation of IQL (Kostrikov et al., 2021) for our offline RL algorithm. The hyperparameters used for the experiment are in Table 2.

Hyperparameter	Value
Offline Sample Count	50000
γ	0.99
Batch Size	256
ψ or Encoder LR	$5 \cdot 10^{-4}$
ψ or Encoder Weight Decay	10^{-4}
ϕ LR	10^{-4}
ϕ Weight Decay	10^{-3}
Actor LR	10^{-4}
Actor β	0.9
Actor logstd Bounds	$[-10, 2]$
Critic LR	10^{-4}
Critic β	0.9
Critic τ	0.005
Quantile	0.7
Optimizer	Adam
Adam β_1	0.9

Table 2. GCB Hyperparameters.

F. Environment Details

We evaluate our method on two Sawyer manipulation environment domains: `Drawer` and `Button and Drawer`. We list details of how we constructed our environments for the PyBullet simulation tasks in Section 7. In the following paragraphs we describe specific details about how we collected data, evaluated the agent, and formed analogies.

For each episode of data collection, we first initialize the PyBullet environment with at most four random objects spawned with random poses in addition to a randomly chosen drawer pose and openness such that the drawer handle, when fully open, fits the xy constraints of the environment. We then roll out a scripted policy $\pi^*(s, g) = a^*$ to complete its task until the goal is achieved or until the 75 timestep limit is reached. In the `Drawer` environment, the task is to open or close the drawer to randomly chosen openness. In the `Button and Drawer` environment, the task is to either do the same task as in the `Drawer` environment or to press a button.

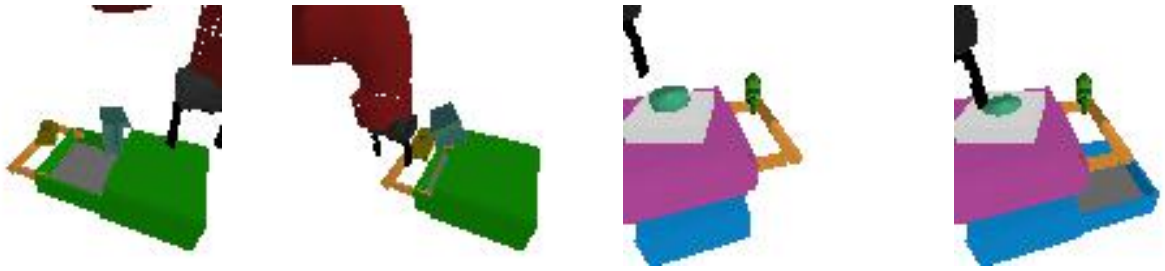
For each episode of evaluation, we randomly initialize the PyBullet environment in the same manner as in data collection. The goal g is collected by rolling out a scripted policy $\pi^*(s, g) = a^*$, rendering the final frame, and marking that as the goal. We then reinitialize the environment with the same object poses for evaluating an agent conditioned on g.

The `Drawer` environment tasks a robotic agent to open or close a drawer to a specific drawer openness. Specifically, it tasks the agent to guide its state image s to a drawer handle and position it as shown in the goal image g using its arm. The environment consists of a drawer with a random pose and at most four random distractor objects.

The `Button and Drawer` environment tasks a robotic agent to either press a button or guide a drawer handle to a specific drawer openness. The environment consists of a button on top of two drawers stacked on top of one another along with at most four random distractor objects. The top drawer is taken from the `Drawer` environment, and the bottom drawer is opened or closed by pressing the button.

To form analogous state goal pairs s_a, g_a, we first randomly initialize the objects in the the `Button and Drawer` environment. We then take the state and add noise to the pose of the drawer (translation and rotation). The colors of the drawer, number and type of distractor objects spawned, and the video distractor in the background of (s_a, g_a) will most likely change as well.

In Table 3 and Table 4, we provide the environment parameter settings we used on each environment. We also provide some extra screenshots of states and goals. For the table below, let $x_p^{(1)}$ denote the top drawer handle xyz position, $d_p^{(1)}$ denote the desired top drawer handle xyz position, $x_p^{(2)}$ denote the bottom drawer handle xyz position, $d_p^{(2)}$ denote the desired bottom drawer handle xyz position, $x_p^{(3)}$ denote the button z position, and $d_p^{(3)}$ denote the desired button z position.



(a) `Drawer` initial state (b) `Drawer` final state (c) `Drawer and Button` initial state (d) `Drawer and Button` final state

Figure 15. Sample images from `Drawer` and `Drawer and Button` environment.

Parameter	Value
Discount γ	0.99
Action Space	$[-1, 1]^5$
Action Space Description	[x, y, z, yaw rotation, claw grip]
DoF	5
Observation Space (i.e. Image Space)	$[0, 1]^{64\times64\times3}$
Max Time Steps	75
Offline Dataset Size	50K transitions
Reward	$\mathbb{1}_{\lvert x_p^{(1)} - d_p^{(1)}\rvert \leq 0.05}$
Number of Distractor Objects	Uniform Random in [0, 4]
Underlying State Space (Not shown to agent)	$\mathbb{R}^{11}$
Underlying State Space Description (Not shown to agent)	[gripper xyz position, gripper quaternion, gripper tips distance, top drawer xyz position]

Table 3. Parameters used in the `Drawer` environment.

Parameter	Value
Discount γ	0.99
Action Space	$[-1, 1]^5$
Action Space Description	[x, y, z, yaw rotation, claw grip]
DoF	5
Observation Space (i.e. Image Space)	$[0, 1]^{64\times64\times3}$
Max Time Steps	75
Offline Dataset Size	50K transitions
Reward	$\begin{cases} \mathbb{1}_{\lvert x_p^{(1)} - d_p^{(1)}\rvert \leq 0.05} & \text{if doing drawer task} \\ \mathbb{1}_{\lvert x_p^{(2)} - d_p^{(2)}\rvert \leq 0.05 \wedge \lvert x_p^{(3)} - d_p^{(3)}\rvert \leq 0.008} & \text{if doing button task} \end{cases}$
Number of Distractor Objects	Uniform Random in [0, 4]
Underlying State Space (Not shown to agent)	$\mathbb{R}^{15}$
Underlying State Space Description (Not shown to agent)	[gripper xyz position, gripper quaternion, gripper tips distance, top drawer xyz position, bottom drawer xyz position, button z position]

Table 4. Parameters used in the `Drawer and Button` environment.