
Learning Plannable Representations with Causal InfoGAN

Thanard Kurutach*
UC Berkeley

Aviv Tamar*
UC Berkeley

Ge Yang
University of Chicago

Stuart Russell
UC Berkeley

Pieter Abbeel
UC Berkeley

Abstract

In recent years, deep generative models have been shown to ‘imagine’ convincing high-dimensional observations such as images, audio, and even video, learning directly from raw data. In this work, we ask how to imagine *goal-directed visual plans* – a plausible sequence of observations that transition a dynamical system from its current configuration to a desired goal state, which can later be used as a reference trajectory for control. We focus on systems with high-dimensional observations, such as images, and propose an approach that naturally combines representation learning and planning. Our framework learns a generative model of *sequential observations*, where the generative process is induced by a transition in a low-dimensional *planning model*, and an additional noise. By maximizing the mutual information between the generated observations and the transition in the planning model, we obtain a low-dimensional representation that best explains the causal nature of the data. We structure the planning model to be compatible with efficient planning algorithms, and we propose several such models based on either discrete or continuous states. Finally, to generate a visual plan, we project the current and goal observations onto their respective states in the planning model, plan a trajectory, and then use the generative model to transform the trajectory to a sequence of observations. We demonstrate our method on imagining plausible visual plans of rope manipulation.

1 Introduction

For future robots to perform general tasks in unstructured environments such as homes or hospitals, they must be able to reason about their domain and plan their actions accordingly. In AI literature, this general problem has been investigated under two main paradigms – automated planning and scheduling [41] (henceforth, AI planning) and reinforcement learning [48] (RL).

Classical work in AI planning has drawn on the remarkable capability of humans to perform long-term reasoning and planning by using abstract representations of the world. For example, humans might think of “cup on table” as a state rather than detailed coordinates or a precise image of such a scene. Interestingly, powerful classical planners exist that can reason very effectively with these kinds of representations, as demonstrated by results in the International Planning Competition [52]. However, such logical representations of the world can be difficult to specify correctly. As an example, consider designing a logical representation for the state of a deformable object such as a rope. Moreover, logical representations that are not grounded *a priori* in real-world observation require a perception module that can identify, for example, exactly when the cup is considered “on the table”. Indeed,

*equal contribution, correspondence to {thanard.kurutach, avivt}@berkeley.edu

most planning successes to date (e.g., [35, 25, 47]) relied on a human-designed state representation, and manually designed the perception of the state from the observation.

In RL, on the other hand, a task is solved directly through trial and error experimentation, guided by a manually provided reward signal. Recent advances in RL using deep neural networks (e.g., [33, 28]) have shown remarkable success in learning policies that act directly on high-dimensional perceptual inputs, such as raw images. Designing a reward function that depends on high-dimensional observations can be challenging, however, and most recent studies either relied on domains where the reward can be instrumented [33, 28, 40], or required successful demonstrations as guidance [16, 45]. Moreover, since RL is guided by the reward to solve a particular task, it does not automatically generalize to different tasks [50, 21]. Recent approaches that aim to achieve generalization in RL through learning on a variety of different tasks (e.g., [12, 53, 14]) are typically not sample-efficient and are limited to relatively simple decision-making problems.

In principle, model-based RL approaches can solve the generalization problem by learning a model of the environment dynamics and planning in that model. However, applying model-based RL to domains with high-dimensional observations has been challenging [55, 17, 15]. Deep learning approaches to learning dynamics models (e.g., action-conditional video prediction models [36, 1, 15]) tend to get bogged down in pixel-level detail, tend to be computationally expensive, and are far from accurate over longer time scales. Moreover, the representations learned using such approaches are typically unstructured, high-dimensional continuous vectors, which cannot be used in efficient planning algorithms. Indeed, prior work has used myopic or random-search-based action selection for planning [1, 15], which can be effective for planning simple skills such as pushing an object to a target, but does not scale up to more complex, high-level decision making problems such as laying the table for dinner.

In this work, we aim to combine the merits of deep learning dynamics models and classical AI planning, and propose a framework for long-term reasoning and planning that is grounded in real-world perception. We present **Causal InfoGAN**, a method for learning *plannable representations* of dynamical systems with high-dimensional observations such as images. By *plannable*, we mean representations that are structured in such a way that makes them amenable for efficient search, through AI planning tools. In particular, we focus on discrete and deterministic dynamics models, which can be used with graph search methods, and on continuous models where planning is done by linear interpolation, though our framework can be generalized to other model types.

In our framework, a generative adversarial net (GAN; [18]) is trained to generate sequential observation pairs from the dynamical system. The generative network (GAN generator) is structured as a deep neural network that takes as input both unstructured random noise and a structured pair of consecutive states from a low-dimensional, parametrized dynamical system termed the *planning model*. The planning model is meant to capture the features that are *most essential for representing the causal properties* in the data, and are therefore important for planning future outcomes. If the planning model is compliant with efficient planning algorithms and is also informative about the high-dimensional observation sequences, then planning using it should be both computationally efficient and also relevant to planning in the actual system we care about. To induce such an informative model, we follow the InfoGAN idea [7], and add to the GAN training a loss function that maximizes the mutual information between the observation pairs and the transitions that induced them.

We train a causal InfoGAN model using data from random exploration in the system. After learning, given an observation of an initial configuration and a goal configuration, we use our model to generate a “walkthrough” sequence of feasible observations that lead from the initial state to the goal. We do this by computing a trajectory in the planning model and using the GAN to transform this trajectory into a sequence of observations. This walkthrough, which breaks the long-horizon planning into a sequence of short-horizon skills, can be later used as a guiding signal for executing the task in the real system.

We compare the representations learned in Causal InfoGAN to standard methods for state aggregation on synthetic tasks, and demonstrate that Causal InfoGAN can generate convincing walkthrough sequences for manipulating a rope into a given shape, using real image data collected by Nair et al. [34] of a robot randomly poking the rope.

2 Preliminaries and Problem Formulation

In this section we present background material and our problem formulation.

2.1 Deep Generative Models based on GAN and InfoGAN

Let $o \sim P_{\text{data}}(o)$ denote observations sampled from a dataset. Deep generative models aim to learn stochastic neural networks that approximate P_{data} . In this work we build on the GAN framework [18], which is composed of a generator, $o = G(z)$, mapping a noise input $z \sim P_{\text{noise}}(z)$ to an observation, and a discriminator, $D(o)$, mapping an observation to the probability that it was sampled from the real data and not from the generator. The GAN training optimizes a game between the generator and discriminator,

$$\min_G \max_D V(G, D) = \min_G \max_D \mathbb{E}_{o \sim P_{\text{data}}} [\log D(o)] + \mathbb{E}_{z \sim P_{\text{noise}}} [\log (1 - D(G(z)))].$$

One can view the noise vector z in the GAN as containing some representation of the observation o . In a general GAN training, however, there is no incentive for this representation to display any structure at all, making it difficult to interpret, or use in a downstream task. The InfoGAN method [7] aims to mitigate this issue.

Let H denote the entropy of a random variable $H(x) = \mathbb{E}_x [-\log(P(x))]$. The mutual information between the two random variables, $I(x; y) = H(x) - H(x|y) = H(y) - H(y|x)$, measures how much knowing one variable reduces the uncertainty about the other variable.

The idea in InfoGAN is to add to the generator input an additional ‘state’² component $s \sim P(s)$, and add to the GAN objective a loss that induces maximal mutual information between the generated observation and the state. The InfoGAN objective is given by:

$$\min_G \max_D V(G, D) - \lambda I(s; G(z, s)), \quad (1)$$

where $\lambda > 0$ is a weight parameter, and $V(G, D)$ is the GAN loss above. Intuitively, this objective induces the state to capture the most salient properties of the observation.

Optimizing the objective in (1) directly is difficult without access to the posterior distribution $P(s|o)$, and a variational lower bound was proposed in [7]. Define an auxiliary distribution $Q(s|o)$ to approximate the posterior $P(s|o)$. Then:

$$I(s; G(z, s)) \geq \mathbb{E}_{s \sim P(s), o \sim G(z, s)} [\log Q(s|o)] + H(s).$$

Using this bound, the InfoGAN objective (1) can be optimized using stochastic gradient descent. Note that the bound is tight when $Q(s|o)$ converges to the true posterior $P(s|o)$.

2.2 Problem Formulation

We consider a fully observable and deterministic dynamical system, $o_{t+1} = f(o_t, u_t)$, where o_t and u_t denote the observation and action at time t , respectively. The function f is assumed to be unknown. We are provided with data $\mathcal{D}$ in the form of N trajectories of observations $\{o_1^i, u_1^i \dots, o_{T_i}^i\}_{i \in 1, \dots, N}$, generated from f , where the actions are generated by an arbitrary exploration policy.³ A typical goal-directed planning problem is the following (e.g., [15, 1]):

Problem 1 Path Planning: Given $\mathcal{D}$, and two observations $o_{\text{start}}, o_{\text{goal}}$, generate a sequence of actions that transition the dynamical system f from o_{start} to o_{goal} .

For realistic long-horizon planning, however, Problem 1 can be unnecessarily difficult to solve. As an example, consider a robot planning to navigate through a building. Planning each motor command for the robot in advance seems redundant – instead, planning a set of way points for the robot and later designing a simple feedback controller to reach them seems much more effective. This concept

²In [7], s is referred to as a *code*. Here we term it as a state, to correspond with our subsequent development of structured GAN input from a dynamical system.

³In this work, we do not concern the problem of how to best generate the exploration data.

of *temporal abstraction* has been fundamental in AI planning (e.g., [13, 49]). To facilitate temporal abstraction in our setting we propose to solve the following, relaxed, problem.

We say that two observations o, o' are *h-reachable* if there exists a sequence of actions that takes the system from o to o' in h or fewer time steps. We consider the problem of generating a *walkthrough* – a sequence of *h-reachable* observations along a feasible path between the start and the goal:

Problem 2 Walkthrough Planning: Given $\mathcal{D}$, h , and two observations o_{start}, o_{goal} , generate a sequence of observations $o_{start}, \dots, o_{goal}$ such that every two consecutive observations in the sequence are *h-reachable*. If such a sequence does not exist, return $\emptyset$.

The motivation to solve problem 2 is that it breaks the long horizon planning problem (from o_{start} to o_{goal}) into a sequence of short h -horizon planning problems which can be later solved effectively using other methods such as inverse dynamics or model-free RL [34]. Note that we are not searching for action sequences, but for a sequence of way point observations. Thus, the actions are not relevant for our problem, and in the sequel we omit them from the discussion.

3 Causal InfoGAN

A natural approach for solving the walkthrough planning problem in Section 2 is learning some model of the dynamics f from the data, and searching for a plan within that model. This leads to a trade-off. On the one hand, we want to be expressive, and learn all the transitions possible from every o within a horizon h . When o is a high dimensional image observation, this typically requires mapping the image to an extensive feature space [36, 15]. On the other hand, however, we want to plan efficiently, which generally requires either low dimensional state spaces or well-structured representations. We approach this challenge by proposing *Causal InfoGAN* – an expressive generative model with a structured representation that is compatible with planning algorithms. In this section we present the Causal InfoGAN generative model, and in Section 4 we explain how to use the model for planning.

Let o and o' denote a pair of sequential observations from the dynamical system f , and let $P_{\text{data}}(o, o')$ denote their probability, as displayed in the data $\mathcal{D}$. We posit that a generative model that can accurately learn $P_{\text{data}}(o, o')$ has to capture the features that are important for representing the *causality* in the data. By causality here, we mean the next observations o' that are reachable from the current observation o . Naturally, such features would be useful later for planning.

We build on the GAN framework [18]. Applied to our setting, a vanilla GAN would be composed of a generator, $o, o' = G(z)$, mapping a noise input $z \sim P_{\text{noise}}(z)$ to an observation pair, and a discriminator, $D(o, o')$, mapping an observation pair to the probability that it was sampled from the real data $\mathcal{D}$ and not from the generator. One can view the noise vector z in such a GAN as a feature vector, containing some representation of the transition to o' from o . The problem, however, is that the structure of this representation is not necessarily easy to decode and use for planning. Therefore, we propose to design a generator with a *structured* input that can be later used for planning. In particular, we propose a GAN generator that is driven by states sampled from a parametrized dynamical system.

Let $\mathcal{M}$ denote a dynamical system with state space S , which we term the set of *abstract-states*, and a parametrized, stochastic transition function $T_{\mathcal{M}}(s'|s)$: $s' \sim T_{\mathcal{M}}(s'|s)$, where $s, s' \in S$ are a pair of consecutive abstract states. We denote by $P_{\mathcal{M}}(s)$ the prior probability of an abstract state s . We emphasize that the abstract state space S can be different from the space of real observations o . For reasons that will become clear later on, we term $\mathcal{M}$ as the *latent planning system*.

We propose to structure the generator as taking in a pair of consecutive abstract states s, s' in addition to the noise vector z . The GAN objective in this case is therefore (cf. Section 2):

$$V(G, D) = \mathbb{E}_{o, o' \sim P_{\text{data}}} [\log D(o, o')] + \mathbb{E}_{z \sim P_{\text{noise}}, s \sim P_{\mathcal{M}}(s), s' \sim T_{\mathcal{M}}(s)} [\log (1 - D(G(z, s, s')))]. \quad (2)$$

The idea is that s and s' would represent the abstract features that are important for understanding the causality in the data, while z would model variations that are less informative, such as pixel level details. To induce learning such representations, we follow the InfoGAN method [7], and add to the GAN objective a loss that induces a maximal mutual information between the generated pair of observations and the abstract states.

We propose the Causal InfoGAN objective:

$$\begin{aligned} \min_{\mathcal{M}, G} \max_D \quad & V(G, D) - \lambda I(s, s'; o, o'), \\ \text{s.t.} \quad & o, o' \sim G(z, s, s') \quad s \sim P_{\mathcal{M}}, \quad s' \sim T_{\mathcal{M}}(s), \end{aligned} \quad (3)$$

where $\lambda > 0$ is a weight parameter, and $V(G, D)$ is given in (2). Intuitively, this objective induces the abstract model to capture the most salient possible changes that can be effected on the observation.

Optimizing the objective in (3) directly is difficult, since we do not have access to the posterior distribution, $P(s, s'|o, o')$, when using an expressive generator function. Following InfoGAN [7], we optimize a variational lower bound of (3). Define an auxiliary distribution $Q(s, s'|o, o')$ to approximate the posterior $P(s, s'|o, o')$. We have, following a similar derivation to [7]:

$$I((s, s'); G(z, s, s')) \geq \mathbb{E}_{\substack{s \sim P_{\mathcal{M}}, s' \sim T_{\mathcal{M}}(s), \\ o, o' \sim G(z, s, s')}} [\log Q(s, s'|o, o')] + H(s, s') \doteq I_{VLB}(G, Q). \quad (4)$$

In this formulation, Q can be seen as a classifier, mapping pairs of observations to pairs of abstract states.

We now note a subtle point. The mutual information in (3) is not sensitive to the order of the code words of the random variables s and s' .⁴ This points to a potential caveat in the optimization objective (3): we would like the random variable for the next abstract state s' to have the *same meaning* as the random variable for the abstract state s . That would allow us to roll-out a sequence of changes to the abstract state, by applying the transition operator $T_{\mathcal{M}}$ sequentially, and effectively plan in the abstract model $\mathcal{M}$. We solve this problem by proposing the *disentangled posterior approximation*, $Q(s, s'|o, o') = Q_1(s|o)Q_2(s'|o')$, and choose $Q_1 = Q_2 \doteq Q$. This effectively induces a generator for which $P(s|o) = P(s'|o')$.⁵

We use the lower bound (4) in (3) to obtain the following loss function:

$$\min_{G, Q, \mathcal{M}} \max_D \quad V(G, D) - \lambda I_{VLB}(G, Q), \quad (5)$$

where $\lambda > 0$ is a constant. The loss in (5) can be optimized effectively using stochastic gradient descent, and we provide a detailed algorithm in Appendix A.

4 Planning with Causal InfoGAN models

In the previous section, we proposed a general framework for learning a deep generative model of data from a dynamical system, with a structured latent space. In this section, we discuss how to use the Causal InfoGAN model for planning goal directed trajectories. We first present our general methodology, and then propose several model configurations for which (5) can be optimized efficiently using backpropagation and the reparametrization trick [7], and the latent planning system is compatible with efficient planning algorithms. We then describe how to combine these ideas for solving the walkthrough planning problem in various domains.

4.1 General Planning Paradigm

Our general paradigm for goal directed planning is described in Figure 1b. We start by training a Causal InfoGAN model from the data, as described in the previous section. Then, we perform the following 3 steps, which are detailed in the rest of this section:

1. Given a pair of observations o_{start}, o_{goal} , we first encode them into a pair of corresponding states s_{start}, s_{goal} . This is described in Section 4.2.
2. Then, using the transition probabilities in the planning model $\mathcal{M}$, we plan a trajectory $s_{start}, s_1, \dots, s_m, s_{goal}$ – a feasible sequence of states from s_{start} to s_{goal} . This is described in Section 4.3.

⁴This is a general property of the entropy of a random variable, which only depends on the probability distribution and not on the variable values. In our case, for example, one can apply a permutation to the transition operator $T_{\mathcal{M}}$, and an inverse of that permutation to the generator’s s' input. Such a permutation would change the meaning of s' , without changing the mutual information term nor the distribution of generated observations.

⁵Note that in a system where the state is fully observable, the posterior is disentangled by definition, therefore in such cases the bound is tight.

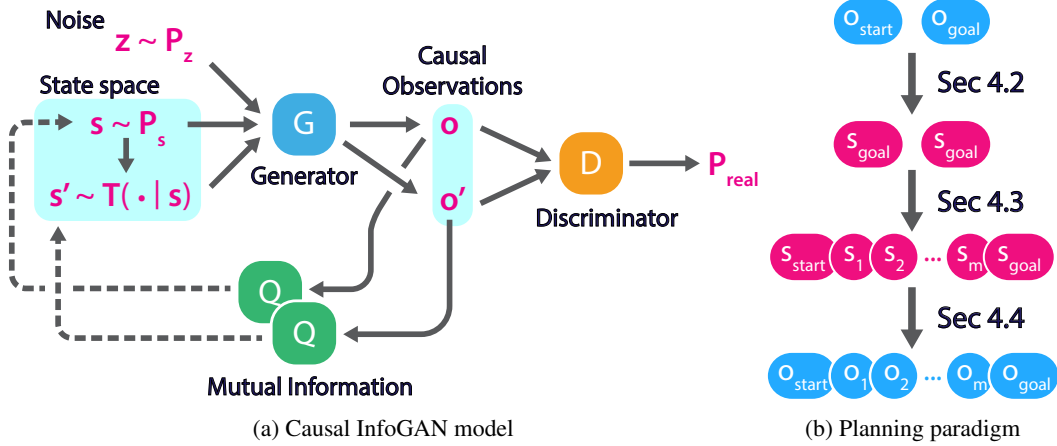


Figure 1: The Causal InfoGAN framework. **(a) Generative model (cf. Section 3).** First, an abstract state s is sampled from a prior $P_{\mathcal{M}}(s)$. Given s , the next state s' is sampled using the transition model $T_{\mathcal{M}}(s'|s)$. The states s, s' are fed, together with a random noise sample z , into the generator which outputs o, o' . The discriminator D maps an observation pair to the probability of the pair being real. Finally, the approximate posterior Q maps from each observation to the distribution of the state it associates with. The causal InfoGAN loss function in Equation (5) encourages Q to predict each state accurately from each observation. **(b) Planning paradigm (cf. Section 4).** Given start and goal observations, we first map them to abstract states, and then we apply planning algorithms using the model $\mathcal{M}$ to search for a path from s_{start} to s_{goal} . Finally, from the plan in abstract states, we generate back a sequence of observations.

3. Finally, we decode the state trajectory into a corresponding trajectory of observations $O_{start}, O_1, \dots, O_m, O_{goal}$. This is described in Section 4.4.

In order for the planned trajectory to be consistent with Problem 2, any two consecutive observations that correspond to consecutive abstract-states, i.e., states that can be reached in a single transition, have to be h -reachable. To train such h -reachable abstract states, we simply train the Causal InfoGAN model with pairs of observations o, o' from $\mathcal{D}$ that are separated by at most h time steps.

The specific method for each step in the planning paradigm can depend on the problem at hand. For example, some systems are naturally described by discrete abstract states, while others are better described by continuous states. In the following, we describe several models and methods that worked well for us, under the general planning paradigm described above. This list is by no means exhaustive. On the contrary, we believe that the Causal InfoGAN framework provides a basis for further investigation of deep generative models that are compatible with planning.

4.2 Encoding an Observation to a State

For mapping an observation to a state, we can simply use the disentangled posterior $Q(s|o)$. We found this approach to work well in low-dimensional observation spaces. However, for high-dimensional image observations we found that the learned $Q(s|o)$ was accurate in classifying generated observations (by the generator), but inaccurate for classifying real observations. This is explained by the fact that in Causal InfoGAN, Q is only trained on generated observations, and can therefore overfit to generated images.

In high-dimensional domains, we therefore opted for a different approach. Following [54], we performed a search over the latent space to find the best latent state mapping $s^*(o)$:

$$s^*(o) = \arg \min_s \min_{s', z} \|o - G(s, s', z)\|^2.$$

Another approach, which could scale to complex image observations, is to add to the GAN training an explicit encoding network [11, 58]. In our experiments, the simple search approach worked well and we did not require additional modifications to the GAN training.

Type	Values s	Prior $P_{\mathcal{M}}(s)$	Transition $T_{\mathcal{M}}(s' s)$	Planning algorithms
Discrete – one-hot	$[N]$	$\mathcal{U}\{1, \dots, N\}$	$s' \sim \text{Softmax}(s^\top \theta)$	Dijkstra
Discrete – binary	$\{0, 1\}^N$	$\mathcal{U}\{0, 1\}^N$	See eq. 6	Dijkstra
Continuous	$\mathbb{R}^N$	$\mathcal{U}(-1, 1)^N$	$s' \sim \mathcal{N}(s, \Sigma_\theta(s))$	Linear interpolation

Table 1: Different models for the latent planning system. In all cases, N is the state dimension. The parameters θ of the transition $T_{\mathcal{M}}$ have different forms depending on the state types. In the one-hot case, θ is a matrix in $\mathbb{R}^{N \times N}$. In the binary case, θ denotes parameters in a stochastic neural network; see Eq. (6). In the continuous case θ represents the parameters of a neural network that controls the variance of the transition.

4.3 Latent Planning Systems

We now present several latent planning systems that are compatible with efficient planning algorithms. Table 1 summarizes the different models.

4.3.1 Discrete Abstract States – One-Hot Representation

We start from a simple abstract state representation, in which each $s \in S$ is represented as a N –dimensional one-hot vector. We denote by $\theta \in \mathbb{R}^{N \times N}$ the model parameters, and compute transition probabilities as: $T_{\mathcal{M}}(s'|s) = \text{Softmax}(s^\top \theta)$. Optimizing the parameters θ with respect to the expectation in the loss (4) is done using the Gumbel-softmax reparametrization trick [19].

4.3.2 Discrete Abstract States – Binary Representation

We now present a more expressive abstract state representation using binary states. Binary state representations are common in AI planning, where each binary element is known as a *predicate*, and corresponds to a particular property of an object being true or false [41]. The Causal InfoGAN framework allows us to learn the predicates *directly from data*.

We propose a parametric transition model that is suitable for binary representations. Let $s \in \{0, 1\}^N$ be an N –dimensional binary vector, drawn from $P_{\mathcal{M}}(s)$. We generate the next state s' by first drawing a random *action vector* $a \in \{0, 1\}^M$ with some probability $P_{\mathcal{M}}(a)$. The purpose of this random action is to generate stochasticity in the state transition. Let $MLP(s, a)$ denote a multi-layered perceptron with parameters θ mapping the state s and action a to $\mathbb{R}^N$. The probability of the next state s' is finally given by:

$$T_{\mathcal{M}}(s' = v|s) = \mathbb{E}_a \left[\prod_i T_{\mathcal{M}}(s'_i = v_i|s, a) \right], \text{ where } T_{\mathcal{M}}(s'_i = 1|s, a) = \text{Sigmoid}(MLP(s, a)_i). \quad (6)$$

Thus, for a given action, each element in $MLP(s, a)$ can be interpreted as the logit in a binary distribution for generating the corresponding element in s' , and for calculating the state transition probability we marginalize over the action. Note that the binary distributions for the different elements in s' are independent *given s and a* . Thus, for a particular s , complex distributions for s' may be expressed through the MLP dependence on a . We further emphasize that there is not necessarily any correspondence between the action vector a and the real actions that generated the observation pairs in the data. The action a is simply a means to induce stochasticity to the state transition network. Optimizing the parameters θ with respect to the expectation in the loss (4) is done using the Gumbel-softmax trick [19] for each element in the MLP output. In this work, we chose $P_{\mathcal{M}}(s)$ and $P_{\mathcal{M}}(a)$ to be fixed distributions, where each binary element was independent, with a Bernoulli(0.5) distribution. In this case, the marginalization can be calculated in closed form. It is also possible to extend this model to a parametric distribution for $P_{\mathcal{M}}(s)$ and $P_{\mathcal{M}}(a)$, and marginalize using sampling.

Both the one-hot and binary representations defined above can be seen as learning a finite Markov decision process (MDP, [6]) model of the data. In the one-hot case, actions in the MDP are implicit in the Gumbel softmax noise, while in the binary case, they are explicit. This is, in fact, a form of state aggregation [6], and we can think of $Q(s|o)$ as a function that assigns a soft clustering to the observations. In contrast to standard clustering approaches in the literature [44, 31, 46, 4], our method

does not require a metric function on the observation space, nor a value function, which depends on a particular task through the reward function. We illustrate these advantages in our experiments.

For planning with discrete models, we interpret the stochastic transition model $T_{\mathcal{M}}$ as providing the possible state transitions, i.e., for every s' such that $T_{\mathcal{M}}(s'|s) > \epsilon$ there exists a possible transition from s to s' . For planning, we require abstract state representations that are compatible with efficient AI planning algorithms. The one-hot and binary representations above can be directly plugged in to graph-planning algorithms such as Dijkstra’s shortest-path algorithm [41].

4.3.3 Continuous Abstract States

For some domains, such as the rope manipulation in our experiments, a continuous abstract state is more suitable. We consider a model where an $s \in S$ is represented as a N –dimensional continuous vector. Planning in high-dimensional continuous domains, however, is hard in general.

Here, we propose a simple and effective solution: we will learn a latent planning system such that *linear interpolation between states makes for feasible plans*. To bring about such a model, we consider transition probabilities $T_{\mathcal{M}}(s'|s)$ given as Gaussian perturbations of the state: $s' = s + \delta$, where $\delta \sim \mathcal{N}(0, \Sigma_{\theta}(s))$ and Σ_{θ} is a diagonal covariance matrix, and is represented by a MLP with parameters θ . The key idea here is that, if only small local transitions are possible in the system, then a linear interpolation between two states s_{start}, s_{goal} has a high probability, and therefore represents a feasible trajectory in the observation space. To encourage such small transitions, we add an L2 norm of the covariance matrix to the full loss (5).

$$L_{cont}(\mathcal{M}) = \mathbb{E}_{s \sim P_{\mathcal{M}}} \|\Sigma_{\theta}(s)\|_2 \quad (7)$$

The prior probability $P_{\mathcal{M}}$ for each element of s is uniform in $[-1, 1]$. Optimizing the parameters θ with respect to the expectation in the loss (4) is done using the reparametrization trick [24].

4.4 Decoding a State Trajectory to an Observation Walkthrough Trajectory

We now discuss how to generate a feasible sequence of observations from a state trajectory in the latent planning system. Here, as before, we separate the discussion for systems with low-dimensional observations and systems with high-dimensional observations, as we found that different practices work best for each.

For low-dimensional observations, we structure the GAN generator G to have an observation-conditional form:

$$o = G_1(z, s, s'), \quad o' = G_2(z, o, s, s'). \quad (8)$$

Using this generator form, we can sequentially generate observations from a state sequence $s_1, \dots, s_T$. We first use G_1 to generate o_1 from s_1, s_2 , and then, for each $2 \leq t < T$, use G_2 to generate o_{t+1} from s_t, s_{t+1} , and o_t .

For high-dimensional image observations, the sequential generator does not work well, since small errors in the image generation tend to get accumulated when fed back into the generator. We therefore follow a different approach. To generate the i ’th observation in the trajectory o_i , we use the generator with the input s_i, s_{i+1} , and a noise z that is fixed throughout the whole trajectory. The generator actually outputs a pair of sequential images, but we discard the second image in the pair.

To further improve the planning result we generate K random trajectories with different random noise z , and select the best trajectory by using a discriminator D to provide a confidence score for each trajectory. In the low-dimensional case, we use the GAN discriminator. In the high-dimensional case, however, we find that the discriminator tends to overfit to the generator. Therefore, we trained an auxiliary discriminator for novelty detection, as described in the Experiment Section 6.2.

5 Related Work

Combining deep generative models with structured dynamical systems has been explored in the context of variational autoencoders (VAEs), where the latent space was continuous [8, 20]. Watter et al. [55] have suggested to use such models for planning, by learning latent linear dynamics, and using a

linear quadratic Gaussian control algorithm for planning. Disentangled video prediction [10] separates object content and position, but has not been used for planning. Very recently, Corneil et al. [9] suggested Variational State Tabulation (VaST) – a VAE-based approach for learning latent dynamics over binary state representations, and planning in the latent space using prioritized sweeping to speed up RL. Causal InfoGAN shares several similarities with VaST, such as using Gumbel-Softmax to backprop through transitions of discrete binary states, and leveraging the structure of the binary states for planning. However, VaST is formulated to require the agent actions, and is thus limited to single time step predictions. More generally, our work is developed under the GAN formulation, which, to date, has several benefits over VAEs such as superior quality of image generation [22]. Causal InfoGAN can also be used with continuous abstract states.

The semiparametric topological memory (SPTM) [43] is another recent approach for solving problems such as Problem 2, by planning in a graph where every observation in the data is a node, and connectivity is decided using a learned similarity metric between pairs of observations. SPTM has shown impressive results on image-based navigation. However, Causal InfoGAN’s *parametric approach* of learning a compact, model for planning has the potential to scale up to more complex problems, in which the increasing amount of data required would make the nonparametric SPTM approach difficult to apply.

Learning state aggregation and state representation has a long history in RL. Methods such as in [32, 44] exploit the value function for measuring state similarity, and are therefore limited to the task defined by the reward. Methods for general state aggregation have also been proposed, based on spectral clustering [31, 46, 30, 29], and variants of K-means [4]. All these approaches rely in some form on the Euclidean distance as a metric between observation features. As we show in our experiments, the Euclidean distance can be unsuitable even on low-dimensional continuous domains.

Recent work in deep RL explored learning goal-conditioned value functions and policies [3, 38], and policies with an explicit planning computation [50, 37, 45]. These approaches require a reward signal for learning (or supervision from an expert [45]). In our work, we do not require a reward signal, and learn a general model of the dynamical system, which is used for goal-directed planning.

Our work is also related to learning models of intuitive physics. Previous work explored feedforward neural networks for predicting outcomes of physical experiments [27], neural networks for modelling relations between objects [56, 42], and prediction based on physics simulators [5, 57]. To the best of our knowledge, these approaches cannot be used for planning, which is the focus in this paper. However, related ideas would likely be required for scaling our method to more complex domains, such as manipulating several objects.

Using the mutual information as a signal that drives prediction in dynamical systems has also been explored under a different formulation in the information bottleneck line of work [51, 2].

In the planning literature, most studies relied on manually designed state representations. In a recent work, Konidaris et al. [26] automatically extracted state representations from raw observations, but relied on a prespecified set of skills for the task. In our work, we automatically extract state representations by learning salient features that describe the causal structure of the data.

6 Experiments

In our experiments, we aim to (1) visualize the abstract states and planning in Causal InfoGAN; (2) compare Causal-InfoGAN with recent state-aggregation methods in the literature; (3) show that Causal InfoGAN can produce realistic visual plans in a complex dynamical system; and (4) show that Causal InfoGAN significantly outperforms baseline methods.

We begin our investigation with a set of toy tasks, specifically designed to demonstrate the benefits of Causal InfoGAN, where we can also perform an extensive quantitative evaluation. We later present experiments on a real dataset of robotic rope manipulation. Technical details for reproducing the experiments are provided in the supplementary material. Code will be made available online at <http://github.com/thanard/causal-infogan>.

6.1 Illustrative Experiments

In this section we evaluate Causal InfoGAN on a set of 2D navigation problems. These toy problems abstract away the challenges of learning visual features, and allow us to make an informative comparison on the task of learning causal structure in data, and using it for planning. For details of the training data see Appendix B.

Our toy domains involve a particle moving in a 2-dimensional continuous domain with impenetrable obstacles, as depicted in Figures 2 and 3. The observations are the (x, y) coordinates of the particle in the plane, and, in the door-key domain, also a binary indicator for holding the key. We generate data trajectories by simulating a random motion of the particle, started from random initial points. We consider the following various geometrical arrangements of the domain, chosen to demonstrate the properties of our method.

1. **Tunnels:** the domain is partitioned into two unconnected rooms (top/bottom), where in each room there is an obstacle, positioned such that transitioning between the left/right quadrants is through a narrow tunnel.
2. **Door-key:** two rooms are connected by a door. The door can be traversed only if the agent holds the key, which is obtained by moving to the red-marked area in the top right corner of the upper room. Holding the key is represented as a binary 0/1 element in the observation.
3. **Rescaled door-key:** Same as door key domain, but the key observation is rescaled to be a small ϵ when the agent is holding the key, and 0 otherwise.

Our domains are designed to distinguish when standard state aggregation methods, which rely on the Euclidean metric, can work well. In the tunnel domain, the Euclidean metric is not informative about the dynamics in the task – two points in different rooms inside the tunnel can be very close in Euclidean distance, but not connected, while points in the same room can be more distant but connected. In the door-key domain, the Euclidean distance is informative if observations with key and without key are very distant in Euclidean space, as in the 0/1 representation (compared to the domain size which is in $[-1, 1]$). In the rescaled door-key, we make the Euclidean distance less informative by changing the key observation to be $0/\epsilon$.

We compare Causal InfoGAN with several recent methods for aggregating observation features into states for planning. Note that in these simple 2D domains, feature extraction is not necessary as the observations are already low dimensional vectors. The simplest baseline is K-means, which relies on the Euclidean distance between observations. In [4], a variant of K-means for temporal data was proposed, using a window of consecutive observations to measure a smoothed Euclidean distance to a cluster centroids. We refer to this method as temporal K-means. In [31], and more recently [46] and [30], spectral clustering (SC) was proposed to learn connected clusters in the data. For continuous observations, SC requires a distance function to build a connectivity graph, and previous studies [31, 46, 30] relied on the Euclidean distance, either by using nearest neighbor to connect the graph, or by using the exponentiated distance to assign edge weights.

In Figure 2, we show the Causal InfoGAN classification of observations to abstract states, $Q(s|o)$, and compare with the K-means baseline; the other baselines gave qualitatively similar results. Note that Causal InfoGAN learned a clustering that is related to the dynamical properties of the domain, while the baselines, which rely on a Euclidean distance, learned clusters that are not informative about the real possible transitions. As a result, the Causal InfoGAN clearly separates abstract states within each room, while the K-means baseline clusters observations across the wall. This demonstrates the potential of Causal InfoGAN to learn meaningful state abstractions without requiring a distance function in observation space. In Figure 3 we show similar results for the door-key domain. When the key observation was scaled to $0/\epsilon$, standard clustering methods did not separate states with the key and without key to different clusters. Causal InfoGAN, on the other hand, learned a binary predicate for holding the key, and learned that obtaining the key happens in the correct position.

To evaluate planning performance, we hand-coded an oracle function that evaluates whether an observation trajectory is feasible or not (e.g., does not cross obstacles, correctly reports $\emptyset$ when a trajectory does not exist). For causal InfoGAN, we ran the planning algorithm described in Section 4. For baselines, we calculated cluster transitions from the data, and generated planning trajectories in observation space by using the cluster centroids. We chose algorithm parameters and stopping criteria by measuring the average feasibility score on a validation set of start/goal observations, and

report the average feasibility on a held out test set of start/goal observations. We report our results in Table 2. The Causal InfoGAN learned clusters that respect the causal properties of the domain, resulting in significantly better planning.

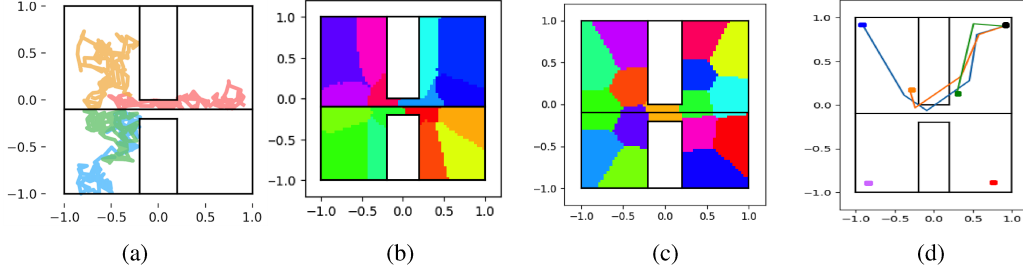


Figure 2: 2D particle results on tunnel domain. (a) The domain - top/bottom rooms are not connected. Left/right quadrants are connected through a narrow tunnel. An example of several random walk trajectories are shown. (b) Clustering found by Causal InfoGAN. (c) Clustering found by K-means. (d) Example walkthrough trajectories generated by Causal InfoGAN, from a point at the top right to five other locations on the map, marked by colored circles. For trajectories that were not found only the target is shown. Note that Causal InfoGAN learned clusters that correspond to the possible dynamics of the particle in the task, and was therefore able to generate reasonable planning trajectories.

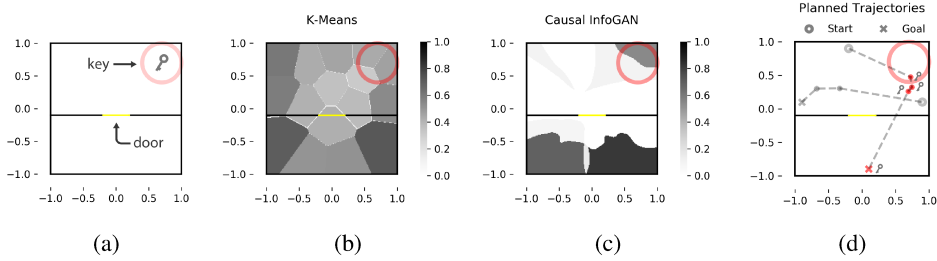


Figure 3: 2D particle results on the ϵ -key domain where the key dimension is scaled down to 0.1. (a) **The key domain**: The rooms are separated in-between by a wall with a door (yellow). The door only opens when the agent has the key, which can be obtained if the agent is within the area indicated by the red circle on the upper right corner. (b) From no-key to has key, **k-means**: Value indicates the probability for the agent to transition from a state not having the key to a state having the key at each (x, y) location. This transition should only occur near the key region (indicated by the red ring). In this case, K-means fails to learn the separation between having and not having the key, and generated high transition probability over the entire domain. (c) The same figure as (b), generated by the **Causal InfoGAN**. On the top right corner where the key is located, the GAN correctly learns that it can transition from having no key to having the key. Bottom blots appear where the posterior sees no data. (d) Causal-InfoGAN planned walkthrough trajectories, showing how the agent acquires the key to cross the door. When the goal is in the top room, the agent goes directly towards the goal without making a detour for the key region.

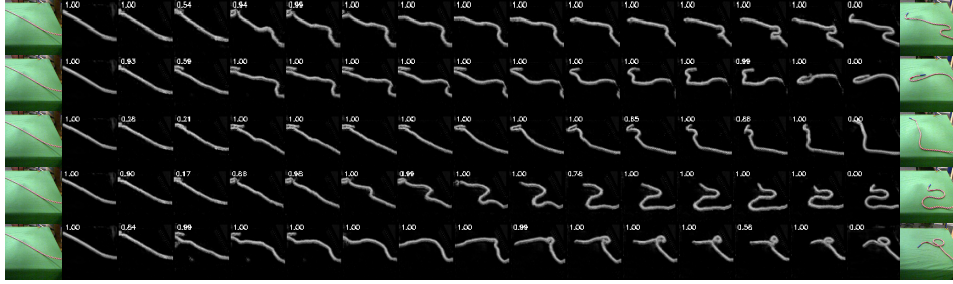
6.2 Rope Manipulation

In this section we demonstrate Causal InfoGAN on the task of generating realistic robotic rope manipulation trajectories from start to goal configurations. Then, we show that Causal InfoGAN generates significantly better trajectories than those generated by the state-of-the-art generative model baselines both visually and numerically.

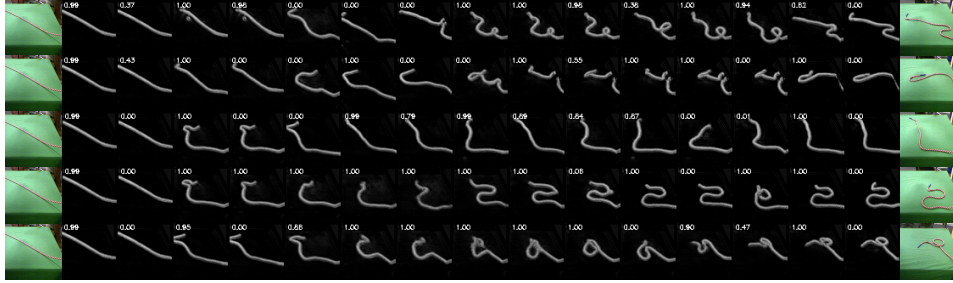
The rope manipulation dataset [34] contains a set of sequential images of a robot manipulating a rope in a self-supervised manner, by randomly choosing a point on the rope and perturbing it slightly. Using this data, the task is to manipulate the rope in a goal-oriented fashion, from one configuration to another, where a goal is represented as an image of the desired rope configuration. In the original study, Nair et al. [34] used the data to learn an inverse dynamics model for manipulating the rope

	Tunnels	Door-key	Rescaled door-key
Causal-InfoGAN	98%	98%	97%
K-means	12.25%	100%	0.0%
Temporal K-means	7.0%	100%	0.0%
Spectral clustering	8.75%	60%	20.0%

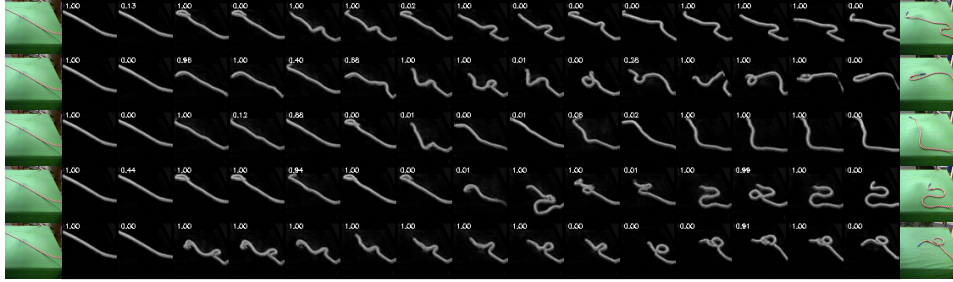
Table 2: Planning results for illustrative 2D tasks. Table shows average feasibility of plans (higher is better) generated by the different algorithms. Note that Causal-InfoGAN significantly outperforms baselines in domains where the Euclidean distance is not informative for planning.



(a) Causal InfoGAN



(b) InfoGAN



(c) DCGAN

Figure 4: Results for rope manipulation data. We compare planning using Causal InfoGAN (top), InfoGAN (middle), and DCGAN (bottom) by interpolation in the latent space, for several rope manipulation goals starting from the same initial configuration. Each plot shows 5 planning instances, from left (starting observation) to the right (goal observation). For each instance, the shown trajectory is picked using the highest trajectory score. The training loss of Causal InfoGAN led to a latent space that is the most accurately represents possible changes to the rope, compared to the other two baselines.

between two images of similar rope configurations. Then, to solve long-horizon planning, Nair et al. required a human to provide the walkthrough sequence of rope poses, and used the learned controller to execute the short-horizon transitions within the plan.

In our experiment, we show that Causal InfoGAN can be used to generate walkthrough plans directly from data for long-horizon tasks, without requiring additional human guidance. We train a Causal-

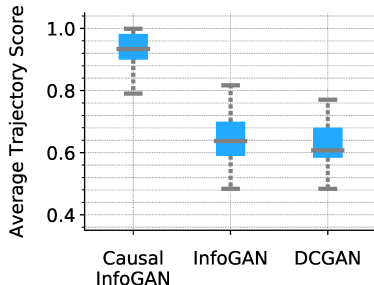


Figure 5: Evaluation of walkthrough planning in rope domain. We trained a classifier to predict whether two observations are sequential or not (1=sequential, 0=not sequential), and compare the average classification score for different generative models. Note that Causal InfoGAN significantly outperforms the baselines, in alignment with the qualitative results of Figure 4.

InfoGAN model on the rope manipulation data of [34]. We pre-processed the data by removing the background, and applying a grayscale transformation. We chose the continuous abstract state representation described in Section 4. In Figure 4a, we show our results for planning walkthroughs between different rope observations. Note that planning here is simply interpolation in the abstract space, however, the Causal InfoGAN objective guarantees that such interpolation indeed relates to feasible transitions. In comparison, in Figure 4b, we trained a standard InfoGAN model, where the mutual information loss does not involve state transitions, and perform interpolation in the InfoGAN abstract state space. We also trained a standard DCGAN model as another baseline, where the observations do not share mutual information with the abstract states, as shown in Figure 4c.⁶ We see that, due to the causality preserving loss, only Causal InfoGAN learns a smooth latent space in which linear interpolation indeed correspond to plausible trajectories in the observation space.

Unlike the synthetic 2D domains above, numerically evaluating planning performance is difficult, since we cannot design a perfect oracle for evaluating the feasibility of a generated visual plan. Instead, we propose a surrogate evaluation criteria: we exploit the fact that we have data of full manipulation trajectories, and train a binary classifier to classify whether two images are sequential in the data or not⁷. For an image pair, the classifier output therefore provides a score between 0 and 1 for the feasibility of the transition. We apply the classifier to compute the *trajectory score* which is the average classifier score of image pairs in the trajectory. Note that this classifier is trained independent of the generative models. Thus, the trajectory score is an impartial metric. For each start and goal, we pick the best trajectory score out of 400 samples of the noise variable z .⁸ As shown in Figure 5, Causal InfoGAN achieved a significantly higher trajectory score averaged over 57 task configurations.

7 Conclusion

We presented Causal InfoGAN, a framework for learning deep generative models of sequential data with a structured latent space. By choosing the latent space to be compatible with efficient planning algorithms, we developed a framework capable of generating goal-directed trajectories from high-dimensional dynamical systems.

Our results for generating realistic manipulation plans of a rope suggest promising applications in robotics, where designing models and controllers for manipulating deformable objects is challenging.

The binary latent models we explored provide a connection between deep representation learning and classical AI planning, where Causal InfoGAN can be seen as a method for learning object predicates directly from data. In future work we intend to investigate this direction further, and incorporate object-oriented models, which are a fundamental component in classical AI.

⁶For DCGAN and InfoGAN, Encoding an observation to a latent state, and decoding a latent state trajectory to an observation walkthrough were done using a similar approach to the Causal InfoGAN method described in Section 4.

⁷The positive data are the pairs of rope images that are 1 step apart and the negative data are randomly chosen pairs that are from different runs which are highly likely to be farther than 1 step apart.

⁸This selection process is applied the same way to the DCGAN and InfoGAN baselines.

References

- [1] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.
- [2] N. Amir, S. Tiomkin, and N. Tishby. Past-future information bottleneck for linear feedback systems. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 5737–5742. IEEE, 2015.
- [3] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.
- [4] N. Baram, T. Zahavy, and S. Mannor. Spatio-temporal abstractions in reinforcement learning through neural encoding. 2016.
- [5] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.
- [6] D. Bertsekas. Dynamic programming and optimal control: Volume II. 2005.
- [7] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.
- [8] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.
- [9] D. Corneil, W. Gerstner, and J. Brea. Efficient model-based deep reinforcement learning with variational state tabulation. *arXiv preprint arXiv:1802.04325*, 2018.
- [10] E. L. Denton and v. Birodkar. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems 30*, pages 4414–4423. 2017.
- [11] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.
- [12] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL²: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.
- [13] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251 – 288, 1972.
- [14] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
- [15] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017.
- [16] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.
- [17] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 512–519. IEEE, 2016.
- [18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [19] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

- [20] M. Johnson, D. K. Duvenaud, A. Wiltchko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, pages 2946–2954, 2016.
- [21] K. Kanksy, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *ICML*, volume 70, pages 1809–1818, 2017.
- [22] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [24] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [25] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos. Next-generation airborne collision avoidance system. Technical report, Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States, 2012.
- [26] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.
- [27] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016.
- [28] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17, 2016.
- [29] M. Liu, M. C. Machado, G. Tesauro, and M. Campbell. The eigenoption-critic framework. *arXiv preprint arXiv:1712.04065*, 2017.
- [30] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956*, 2017.
- [31] S. Mahadevan and M. Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(Oct):2169–2231, 2007.
- [32] S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 71. ACM, 2004.
- [33] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [34] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2146–2153. IEEE, 2017.
- [35] N. J. Nilsson. Shakey the robot. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1984.
- [36] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.
- [37] J. Oh, S. Singh, and H. Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pages 6118–6128, 2017.
- [38] V. Pong, S. Gu, M. Dalal, and S. Levine. Temporal difference models: Model-free deep RL for model-based control. *CoRR*, abs/1802.09081, 2018.

- [39] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [40] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degraeve, T. Van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.
- [41] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
- [42] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4974–4983, 2017.
- [43] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.
- [44] D. I. Simester, P. Sun, and J. N. Tsitsiklis. Dynamic catalog mailing policies. *Management science*, 52(5):683–696, 2006.
- [45] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.
- [46] A. Srinivas, R. Krishnamurthy, P. Kumar, and B. Ravindran. Option discovery in hierarchical reinforcement learning using spatio-temporal clustering. *arXiv preprint arXiv:1605.05359*, 2016.
- [47] S. Srivastava, S. Zilberstein, A. Gupta, P. Abbeel, and S. Russell. Tractability of planning with loops. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3393–3401. AAAI Press, 2015.
- [48] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [49] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.
- [50] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *NIPS*, pages 2146–2154, 2016.
- [51] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.
- [52] M. Vallati, L. Chrapa, M. Grześ, T. L. McCluskey, M. Roberts, S. Sanner, et al. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3):90–98, 2015.
- [53] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016.
- [54] W. Wang, A. Wang, A. Tamar, X. Chen, and P. Abbeel. Safer classification by synthesis. *arXiv preprint arXiv:1711.08534*, 2017.
- [55] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.
- [56] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran. Visual interaction networks. *arXiv preprint arXiv:1706.01433*, 2017.
- [57] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, pages 152–163, 2017.
- [58] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

A Algorithm

Given the training data $\mathcal{D} = \{(o, o') \mid o \text{ and } o' \text{ are sequential observations}\}$, Causal infoGAN learns a generative model that structures the latent space in a way that is useful for planning. We provide the algorithm details below:

Let $\theta_D, \theta_G, \theta_Q, \theta_T$ denote the parameters of neural networks $D, G, Q, T_{\mathcal{M}}$, respectively.

For a minibatch of m samples $\{(o_i, o'_i)\}_{i=1}^m$ from $\mathcal{D}$, we:

- Generate m fake samples
 - Sample abstract states $s_1, \dots, s_m$, where $s_i \sim P_{\mathcal{M}}$
 - Sample next states $s'_1, \dots, s'_m$, where $s'_i \sim T_{\mathcal{M}}(s'_i | s_i)$
 - Sample noise $z_1, \dots, z_m$, where $z_i \sim P_{\text{noise}}$
 - Generate fake observations $\hat{o}_1, \hat{o}'_1, \dots, \hat{o}_m, \hat{o}'_m$, where $\hat{o}_i, \hat{o}'_i = G(z_i, s_i, s'_i)$.
- Update the discriminator by descending its stochastic gradient

$$\nabla_{\theta_D} \left(-\frac{1}{m} \sum_{i=1}^m [\log D(o_i, o'_i) + \log(1 - D(\hat{o}_i, \hat{o}'_i))] \right)$$

- Update the generator and transition model by descending its stochastic gradient

$$\nabla_{\theta_G, \theta_T} \left(-\frac{1}{m} \sum_{i=1}^m [\log D(\hat{o}_i, \hat{o}'_i)] \right),$$

where the gradient θ_T is backpropagated using the reparametrization trick of Gumbel-softmax [19].

- Update posterior, generator, and transition model in the direction of maximal mutual information

$$\nabla_{\theta_Q, \theta_G, \theta_T} \left(\frac{1}{m} \sum_{i=1}^m [\log P_{\mathcal{M}}(s_i) - \log Q(\hat{o}_i | s_i) + \log T_{\mathcal{M}}(s'_i | s_i) - \log Q(\hat{o}'_i | s'_i)] \right),$$

where the gradient θ_T is backpropagated using the reparametrization trick of Gumbel-softmax [19].

- (*For continuous states with linear interpolation planning*) Update transition model to ensure small local transitions in the state space generate plausible observations.

$$\nabla_{\theta_T} \left(\frac{1}{m} \sum_{i=1}^m \|\Sigma_{\theta_T}(s_i)\|_2 \right),$$

where Σ_{θ_T} is part of $T_{\mathcal{M}}$ (see Section 4.3).

- (*Optional*) Update transition model to minimize a self-consistency loss (see details below)

$$\nabla_{\theta_T} \left(\frac{1}{m} \sum_{i=1}^m [-\log T_{\mathcal{M}}(s^*(o'_i) | s^*(o_i))] \right),$$

where $s^*(o) = \arg \max_s Q(s | o)$.

The self-consistency loss is added to further strengthen the relationship between transitions in the latent planning system and the real observations. We maximize the likelihood of observed transitions in the predicted states from real transitions. Namely, let $s^*(o) = \arg \max_s Q(s | o)$ denote the most likely state encoding for an observation, then the self-consistency loss is given by,

$$L_{sc}(\mathcal{M}) = \mathbb{E}_{o, o' \sim P_{\text{data}}} [-\log T_{\mathcal{M}}(s^*(o') | s^*(o))].$$

This loss guides T to be consistent with Q which stabilizes the training. We found this loss to help in stabilizing training for low-dimensional observations. We did not find that adding this loss is beneficial in the high-dimensional case, since in that case, while Q provides meaningful state estimation on fake observations, it tends to overfit to the generated samples, and does not predict reliable states on real observations.

B Experiment details

B.1 2D Navigation Experiment

The model parameters we used for the toy domains is as follows:

In the key domain, we used a 4-dimensional space for the latent state (we also experimented with 3-5 dimensional latent spaces which gave similar results. Smaller latent space tend to have less expressive power and subject to generator collapse. Beyond 5 however the benefit is marginal.) Actions are sampled from a 3 dimensional space whereas the noise z is 4 dimensional. The loss for the generator, the posterior and the transition consistency are weighted equally with a learning rate of 10^{-4} , whereas the learning rate for the discriminator is five times larger (5×10^{-4}). We found that the transition consistency loss important in the stability of models using binary representations. The same hyperparameters are used for both the key domain and the ϵ -key domain. In the latter $\epsilon = 0.1$.

In the tunnel domain we also used a 4-dimensional latent state (a 3-dimensional latent state gave similar results). Actions are sampled from a 3-dimensional space and noise is 4 dimensional. The learning rates are identical to those of the key domain.

To generate the training samples in the tunnel domain, the random walk had a characteristic length scale of 0.05. The rooms are from -1 to 1 in both width and height. The meridian is placed slightly off the middle, at $y=-0.1$. We bias the starting point in the particle trajectories around the choke in the middle, so that the sample trajectories have substantial probability is crossing from one room to the other.

In the key domain, we used a a characteristic length scale of 0.3. This much larger step size is needed because the particle needs to cover the top room, make it to the key zone (to obtain the key), and carry the key to the door to cross to the bottom room in a single trajectory.

In the tunnel domain, we chose the horizon k to be uniform in $5 - 9$. In the key domain, since the charecteristic length scale is larger, we chose k to be in $1 - 4$.

In the key domain, we represent the possession of the key by a single number in the binary set $\{0, 1\}$. Incidentally, it was necessary to inject Gaussian noise to this key dimension during training. Otherwise the generator is required to learn a singularity around 0 and 1, making it numerically highly unlikely. We varied the normalized standard deviation of this Gaussian noise (w.r.t. ϵ). Larger noise (≤ 0.2) produces more stable training, but too much noise can cause blurriness in the cluster boundaries. Overall the scale of this Gaussian noise doesn't substantially impact the representation that is learned.

The models are identical between the key domain and the tunnel domain. Both the generator, the discriminator and the binary posterior are two layer perceptrons with two 100 dimensional hidden layers. The transition function also has two hidden layers, with 10 neurons each.

For the represetantion of the latent space, we use a binary representation of the states as described in section 4.3.2. The generator uses a sequential architecture as described in Section 4.4, but the two outputs of the generator are trained with only 1 timestep in between with no autoregression on the autoregressive sub module.

B.2 Rope Experiment

We use Adam [23] optimizer with the learning rate of 0.0002 for both discriminator and generator losses. The generator loss is the sum of three losses described in the Appendix A. We use coefficients 1 for the main generator loss, and 0.1 for both the mutual information and the transition loss. We deploy standard DCGAN architectures [39] for the discriminator D and the generator G. The posterior estimator Q has the same architecture as D with the change of the last CNN layer to output 128 channels and the addition of another layer of batchnorm, leaky ReLU and conv layer to the dimension of code. The details are described in table 3.

In DCGAN and infoGAN baselines, the size of latent code (or abstract state) and the noise is 7 and 2 respectively. In Causal InfoGAN, the generator takes in two abstract states at the same time so the size of noise is doubled to 4. However, we found that the result is quite robust to the dimension sizes.

The latent planning system uses a uniform prior between $[-1, 1]$ and a Gaussian transition with zero mean and state-dependent variance. The variance is diagonal and parametrized by a two-layer feed forward neural network of size 64 with ReLU nonlinearity. The last layer is exponentiated to output a positive value for the variance.

For training set, we use sequential observation pairs with 1 step apart from the rope dataset by Nair et al. [34].

B.2.1 Causal Classifier

We trained a binary classifier to function as an evaluator for whether an observation transition is feasible or not, given the data. We use the classifier for two tasks: (1) To post-select transitions (in the observation space) during planning, and (2) to evaluate the score of a walkthrough trajectory.

During training, the classifier takes in a pair of images and output a binary classification of whether this image pair appears sequentially related. The training dataset consists of positive image pairs that are 1 timestep apart, and negative pairs that are randomly sampled from different rope manipulation runs. To avoid overfitting to the background in the rope dataset and learning a trivial solution where the classifier uses the background to distinguish different runs, we preprocess the rope data using the background subtraction pipeline mentioned above.

The training accuracy converges to 100% on the training set, and 98% on a held-out test set.

To validate that this classifier actually learns to tell if the transition between two images is feasible or not, we evaluate it on images that are k steps apart where the largest k is the length of an rope experiment. Despite the classifier never seeing samples that are more than 1 step apart, it learns to predict 0 probability for image pairs with large k . The prediction is well-behaved – As we increase k from 1 to the length of the run, the binary output smoothly and monotonically decreases from 1 to 0.

The model architecture used is a convolutional neural network with the following architecture. The two input images are concatenated channel-wise, fed together into the classifier. The optimization is done with the Adam optimizer with a learning rate of 10^{-3} . These hyperparameters are not tuned since the performance of the classifier is sufficient.

discriminator D / posterior estimator Q	generator G
Input 64 x 64 grayscale images (2 for D and 1 for Q)	Input a vector in $\mathbf{R}^7 \times \mathbf{R}^7 \times \mathbf{R}^4$
4 x 4 conv. 64 lReLU, stride 2, batchnorm	4 x4 upconv. 512 lReLU, stride 2, batchnorm
4 x 4 conv. 128 lReLU, stride 2, batchnorm	4 x4 upconv. 256 lReLU, stride 2, batchnorm
4 x 4 conv. 256 lReLU, stride 2, batchnorm	4 x4 upconv. 128 lReLU, stride 2, batchnorm
4 x 4 conv. 512 lReLU, stride 2, batchnorm	4 x4 upconv. 64 lReLU, stride 2, batchnorm
4 x 4 conv. 1 for D and 128-batchnorm-lReLU-7 for Q	4 x4 upconv. 2 Tanh (1 channel for each image)

Table 3: The architectures for generating rope images. The discriminator takes in two grayscale images, and outputs the probability of the pair being real. The posterior shares the same architecture with D except the first and the last layer. It takes in one image, and outputs the mean and the variance of its predicted state. The generator takes in the current and the next abstract states (dim 7), and the noise (dim 4). It outputs the current and the next observations. The leaky coefficient is 0.2.