Robust Locally-Linear Controllable Embedding

Ershad Banijamali
University of Waterloo

Rui Shu
Standford University

Mohammad Ghavamzadeh
DeepMind

Hung Bui
Adobe Research

Ali Ghodsi
University of Waterloo

Abstract

Embed-to-control (E2C) [17] is a model for solving high-dimensional optimal control problems by combining variational auto-encoders with locally-optimal controllers. However, the current E2C model suffers from two major drawbacks: **1)** its objective function does not correspond to the likelihood of the data sequence and **2)** the variational encoder used for embedding typically has large variational approximation error, especially when there is noise in the system dynamics. In this paper, we present a new model for learning *robust* locally-linear controllable embedding (RCE). Our model directly estimates the predictive conditional density of the future observation given the current one, while introducing the bottleneck [11] between the current and future observations. Although the bottleneck provides a natural embedding candidate for control, our RCE model introduces additional specific structures in the generative graphical model so that the model dynamics can be robustly linearized. We also propose a principled variational approximation of the embedding posterior that takes the future observation into account, and thus, makes the variational approximation more robust against the noise. Experimental results demonstrate that RCE outperforms the existing E2C model, and does so significantly in the regime where the underlying dynamics is noisy.

1 Introduction

Model-based locally optimal control algorithms are popular in controlling non-linear dynamical systems with continuous state and action spaces. Algorithms from this class such as differential dynamic programming (DDP) [3], iterative linear quadratic regulator (iLQR) [8], and iterative linear quadratic Gaussian (iLQG) [14] have been successfully applied to a variety of complex control problems [1, 13, 7, 9]. The general idea of these methods is to iteratively linearize the non-linear dynamics around the current trajectory and then use linear quadratic methodology to derive Riccati-like equations to improve the trajectory. However, these methods assume that the model of the system is known and need relatively low-dimensional state representations. These requirements limit their usage in control of dynamical systems from raw sensory data (e.g., image and audio), a scenario often seen in modern reinforcement learning (RL) systems.

Although both model-based RL and methods to find low-dimensional representations that are appropriate for control (see e.g., [2]) have a long history, they have recently witnessed major improvements due to the advances in the field of deep learning. Deep autoencoders [6, 15] have been used to obtain low-dimensional representations for control, and deep generative models have been used to develop new model-based RL algorithms. However, what is desirable in model-based locally optimal control algorithms is a representation that can be used for learning a model of the dynamical system and can also be systematically incorporated into the existing tools for planning and control. One such model is embed to control (E2C) [17]. E2C turns the problem of locally optimal control in high-dimensional non-linear systems into one of identifying a low-dimensional latent space in which we can easily perform locally optimal control. The low-dimensional latent space is learned using a model based on variational autoencoders (VAEs) [5, 10] and the iLQG algorithm [14] is used for locally optimal control.

While the idea of E2C is intriguing, it suffers from two major statistical deficiencies. **Firstly**, to induce the lower-dimensional embedding, at each time step t, E2C models the pair-marginal distribution of two adjacent observations $(\mathbf{x}_t, \mathbf{x}_{t+1})$. As a result, its loss function effectively is the sum over the pair-marginals, which is clearly not the data likelihood for the entire trajectory. Moreover, at every time step t, E2C needs to enforce the consistency between the posterior of the embedding and the predictive distribution of the future embedding by minimizing their KL divergence. These all indicate that the E2C loss is not a lower bound of the likelihood of the data. The practice of modeling the pair-marginal of $(\mathbf{x}_t, \mathbf{x}_{t+1})$ using a latent variable model also imposes a Gaussian prior on the embedding space, which might be in conflict with the locally-linear constraint that we would like to impose. **Secondly**, the variational inference scheme in E2C attempts to approximate the posterior of the latent embedding via a recognition model that does not depend on the future observation $\mathbf{x}_{t+1}$. We believe that this is done out of necessity, so that the locally-linear dynamics can be encoded as a constraint in the original E2C model. In an environment where the future is uncertain (e.g., in the presence of noise or other unknown factors), the future observation carries significant information about the true posterior of the latent embedding. Thus, a variational approximation family that does not take future observation into account, while approximating the posterior, will result in a large variational approximation error, leading to the learning of a sub-optimal model that underperforms, especially when the dynamics is noisy.

To address these issues, we take a more systematic view of the problem. Instead of mechanically applying VAE to model the pair-marginal density, we build on the recent bottleneck conditional density estimator (BCDE) [11] and directly model the predictive conditional density $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$. The BCDE model introduces a bottleneck random variable $\mathbf{z}_t$ in the middle of the information flow from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. While this bottleneck provides a natural embedding candidate for control, these embeddings need to be structured in a way to respect the locally linear constraint of the dynamics. Our proposed model, *robust controllable embedding* (RCE), provides a rigorous answer to this question in the form of a generative graphical model. A key idea is to explicitly treat the reference linearization point in the locally-linear model as an additional random variable. We also propose a principled variational approximation of the embedding posterior that takes the future observation into account and optimizes a variational lower bound of the likelihood of the data sequence. This allows our framework to provide a clean separation of the generative graphical model and the amortized variational inference mechanism (e.g., the recognition model).

After a brief overview of locally linear control and E2C in Section 2, we present our proposed RCE model in Section 3. Unlike E2C, RCE directly models the conditional density of the next observation given the current one via a form of bottleneck conditional density estimators [11]. In Section 3, we first describe the RCE's graphical model in details and then present the proposed variational approximation of the embedding's posterior. In Section 4, we apply RCE to four RL benchmarks from [17] and show that it consistently outperforms E2C in both prediction and planning. Crucially, we demonstrate the robustness of RCE: as the dynamics becomes more noisy, RCE continues to perform reasonably well while E2C's performance degrades sharply.

2 Preliminaries

In this section, we first define the non-linear control problem that we are interested to solve, and then provide a brief overview of stochastic locally optimal control and the E2C model. We also motivate our proposed robust controllable embedding (RCE) model that will be presented in Section 3.

2.1 Problem Formulation

We are interested in controlling the non-linear dynamical systems of the form

$$\mathbf{s}_{t+1} = f_{\mathcal{S}}(\mathbf{s}_t, \mathbf{u}_t) + \mathbf{n}^{\mathcal{S}}, \tag{1}$$

where $\mathbf{s}_t \in \mathbb{R}^{n_s}$ and $\mathbf{u}_t \in \mathbb{R}^{n_u}$ denote the state and action of the system at time step t, $\mathbf{n}^{\mathcal{S}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{n}^{\mathcal{S}}})$ is the Gaussian system noise, and $f_{\mathcal{S}}$ is a smooth non-linear system dynamics. Note that in this case $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t)$ would be the multivariate Gaussian distribution $\mathcal{N}\big(f_{\mathcal{S}}(\mathbf{s}_t, \mathbf{u}_t), \boldsymbol{\Sigma}_{\mathbf{n}^{\mathcal{S}}}\big)$. We assume that we only have access to the high-dimensional observation $\mathbf{x}_t \in \mathbb{R}^{n_x}$ of each state $\mathbf{s}_t$ ($n_s \ll n_x$) and our goal is to learn a low-dimensional latent state space $\mathcal{Z} \subset \mathbb{R}^{n_z}$ ($n_z \ll n_x$) in which we perform optimal control.

2.2 Stochastic Locally Optimal Control

Stochastic locally optimal control (SLOC) is based on the idea of controlling the non-linear system (1), along a reference trajectory $\{\bar{\mathbf{s}}_1, \bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{s}}_H, \bar{\mathbf{u}}_H, \bar{\mathbf{s}}_{H+1}\}$, by transforming it to a time-varying linear quadratic regulator (LQR) problem

$$\begin{aligned} &\min_{\mathbf{u}_{1:T}} \mathbb{E}\left[\sum_{t=1}^{T} \big((\mathbf{s}_t - \mathbf{s}^f)^\top \mathbf{Q}(\mathbf{s}_t - \mathbf{s}^f) + \mathbf{u}_t^\top \mathbf{R}\mathbf{u}_t\big)\right] \\ &\text{s.t} \quad \mathbf{y}_{t+1} = \mathbf{A}_t\mathbf{y}_t + \mathbf{B}_t\mathbf{v}_t, \end{aligned} \tag{2}$$

where $\mathbf{s}^f$ is the final (goal) state, $\mathbf{Q}$ and $\mathbf{R}$ are cost weighting matrices, $\mathbf{y}_t = \mathbf{s}_t - \bar{\mathbf{s}}_t$, $\mathbf{v}_t = \mathbf{u}_t - \bar{\mathbf{u}}_t$, $\bar{\mathbf{s}}_{t+1} =$

$f_{\mathcal{S}}(\bar{\mathbf{s}}_t, \bar{\mathbf{u}}_t)$, $\mathbf{A}_t = \frac{\partial f_{\mathcal{S}}}{\partial \mathbf{s}}(\bar{\mathbf{s}}_t, \bar{\mathbf{u}}_t)$, and $\mathbf{B}_t = \frac{\partial f_{\mathcal{S}}}{\partial \mathbf{u}}(\bar{\mathbf{s}}_t, \bar{\mathbf{u}}_t)$. Eq. 2 indicates that at each time step t, the non-linear system has been locally approximated with a linear system around the reference point $(\bar{\mathbf{s}}_t, \bar{\mathbf{u}}_t)$ as

$$\mathbf{s}_{t+1} \approx f_{\mathcal{S}}(\mathbf{s}_t, \mathbf{u}_t) + \left[\frac{\partial f_{\mathcal{S}}}{\partial \mathbf{s}}(\bar{\mathbf{s}}_t, \bar{\mathbf{u}}_t)\right](\mathbf{s}_t - \bar{\mathbf{s}}_t) \quad (3)$$
$$+ \left[\frac{\partial f_{\mathcal{S}}}{\partial \mathbf{u}}(\bar{\mathbf{s}}_t, \bar{\mathbf{u}}_t)\right](\mathbf{u}_t - \bar{\mathbf{u}}_t).$$

The RHS of Eq. 2 sometimes contains an offset $\mathbf{c}_t$ resulted from the linear approximation and/or noise

$$\mathbf{y}_{t+1} = \mathbf{A}_t \mathbf{y}_t + \mathbf{B}_t \mathbf{v}_t + \mathbf{c}_t. \quad (4)$$

Eq. 4 can be seen as

$$\begin{bmatrix}\mathbf{y}_{t+1}\\ 1\end{bmatrix} = \begin{bmatrix}\mathbf{A}_t & \mathbf{c}_t\\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{y}_t\\ 1\end{bmatrix} + \begin{bmatrix}\mathbf{B}_t\\ 0\end{bmatrix}\mathbf{v}_t,$$

and thus, can be easily transformed to the standard from (2) by adding an extra dimension to the state as

$$\mathbf{y}'_t = \begin{bmatrix}\mathbf{y}_t\\ 1\end{bmatrix}, \quad \mathbf{A}'_t = \begin{bmatrix}\mathbf{A}_t & \mathbf{c}_t\\ 0 & 1\end{bmatrix}, \quad \mathbf{B}'_t = \begin{bmatrix}\mathbf{B}_t\\ 0\end{bmatrix}.$$

Locally optimal actions in Eq. 2 can be computed in closed-form by solving the local LQRs (3) using the value iteration algorithm.

Since the quality of the control depends on the quality of the reference trajectory, SLOC algorithms are usually iterative (e.g., iLQR and iLQG), and at each iteration generate a better reference trajectory. At the abstract level, a SLOC algorithm operates as follows: at each iteration k, a reference trajectory is generated using the current policy $\pi^{(k)}$, the LQR approximation of the non-linear system is computed around this reference trajectory, and finally the next policy $\pi^{(k+1)}$ is computed by solving this LQR. The algorithm stops after a fixed number of iterations, e.g., 100.

As mentioned in Section 2.1, since we do not have access to the true state $\mathbf{s}$, we perform the optimal control in the low-dimensional latent space $\mathbf{z}$ learned from the observations $\mathbf{x}$. Thus, all the $\mathbf{s}$'s in this section should be replaced by $\mathbf{z}$ in the following sections.

2.3 The Embed to Control (E2C) Model

We now return to the assumption that we only observe a finite number of high-dimensional sensory data (e.g., images) $\mathbf{x}_t \in \mathbb{R}^{n_x}$ from the system. We denote the high-dimensional observation sequence by $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Note that the observations are selected such that the sequence $\mathbf{X}$ is Markovian. For example, $\mathbf{x}$ could be a set of buffered observed images of the system that encodes all the information about the past. Depending on the system, this set may have only one or multiple images.

It is clear that direct control in $\mathbb{R}^{n_x}$ is complicated because of its high-dimensional nature. However, when the true underlying state space is low-dimensional, it would be possible to embed the high-dimensional observations in a low-dimensional latent space $\mathcal{Z}$, in a way that the dynamics of the system can be captured by a much simpler model, which can then be used for optimal control. This general strategy is known as embed to control (E2C) [17]. Note that a suitable embedding function is sufficient for model-based control, we do not need to recover the true state $\mathbf{s}_t$.

We denote by $\mathbf{z}_t$ the low-dimensional embedding of $\mathbf{x}_t$. E2C first introduces a new variable $\hat{\mathbf{z}}_{t+1}$ as the result of applying $\mathbf{u}_t$ to the latent dynamics $f_{\mathcal{Z}}$, i.e.,

$$\hat{\mathbf{z}}_{t+1} = f_{\mathcal{Z}}(\mathbf{z}_t, \mathbf{u}_t) + \mathbf{n}^{\mathcal{Z}}_t, \quad (5)$$

where $\mathbf{n}^{\mathcal{Z}}_t$ denotes the transition noise in the latent space. E2C employs the pair $(\mathbf{z}_t, \hat{\mathbf{z}}_{t+1})$ as the latent variables that model the pair-marginal $p(\mathbf{x}_t, \mathbf{x}_{t+1})$. It uses the variational recognition network $q(\mathbf{z}_t|\mathbf{x}_t)$, while forcing $q(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \mathbf{u}_t)$ to be the generative dynamics of Eq. 5. This leads to the following lower bound of the pair-marginal

$$\begin{aligned}&p(\mathbf{x}_t, \mathbf{x}_{t+1}|\mathbf{u}_t) \geq \mathcal{L}^{\text{bound}}_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\\ &\quad= \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_t)q(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t,\mathbf{u}_t)}\big[-\log p(\mathbf{x}_t|\mathbf{z}_t)\\ &\qquad - \log p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1}) + \text{KL}(q(\mathbf{z}_t|\mathbf{x}_t) \parallel p(\mathbf{z}_t))\big] \quad (6)\end{aligned}$$

Local linearization of the dynamics is enforced inside the recognition model $q(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \mathbf{u}_t)$, where mapping from a linearization point $\bar{\mathbf{z}}_t$ to the linearization matrices are estimated via neural networks.

Finally, we want $\mathbf{z}_{t+1}$ to be both the embedding of $\mathbf{x}_{t+1}$ and the result of applying $\mathbf{u}_t$ to $\mathbf{z}_t$. E2C attempts to enforce this temporal consistency criterion by encouraging the distributions of $\hat{\mathbf{z}}_{t+1}$ and the next step embedding $\mathbf{z}_{t+1}$ to be similar (in the KL sense). Enforcing the temporal consistency leads to the modified objective

$$\mathcal{L}_t = \mathcal{L}^{\text{bound}}_t + \lambda \text{KL}\big(q(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \mathbf{u}_t) \parallel q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})\big), \quad (7)$$

where λ is an additional hyperparameter of the model. We note that neither of the two objectives $\sum_t \mathcal{L}^{\text{bound}}_t$ and $\sum_t \mathcal{L}_t$ is a lower bound of the data likelihood $p(\mathbf{X})$. The fact that E2C does not optimize a proper lower bound of the data likelihood has also been observed by [4].

Compared to E2C, our method is based on introducing a graphical model that clearly separates the generative model from the variational recognition model. This enables us to handle noise in the system and avoid heuristic terms in the objective functions that need extra hyperparameter tuning. Furthermore, we can optimize a lower bound on the likelihood of the data sequence using a better-designed recognition model more robust w.r.t. noise. Note that our goal is not to purely obtain the best predictive power as in [4], but to design a predictive model that yields a suitable embedding representation for locally optimal control. Unlike [4] which does not report control performance, our experiments focus on the performance of the controller under various noise regime. In the next section, we describe our proposed RCE model and demonstrate how it addresses the aforementioned issues of E2C.

3 Model Description

In this section, we first introduce our graphical model that represents the relation between the observations and latent variables in our model. We then derive a lower bound on the likelihood of the observation sequence. The objective of training in our model is to maximize this lower bound. Finally, we describe the details of the method we use for planning in the latent space learned by our model.

3.1 Graphical Model

We propose to learn an action-conditional density model of the observations $\mathbf{x}_{1:N}$. Similar to E2C, we assume that the observation sequence is Markovian. Thus, optimizing the likelihood $p(\mathbf{x}_{1:N}|\mathbf{u}_{1:N})$ reduces to learning an action-conditional generative model that can be trained via maximum likelihood, i.e.,

$$\max_{\theta} \log p_\theta(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t), \tag{8}$$

where the prediction of the next observation $\mathbf{x}_{t+1}$ depends only on the current $\mathbf{x}_t$ and action $\mathbf{u}_t$. Note that our generative model is parameterized by θ. For notational simplicity, we shall omit θ in our presentation.

We first discuss how to learn a low-dimensional representation of $\mathbf{x}$ that adheres to globally linear dynamics by incorporating several constraints into the structure of our generative model. First, we introduce the latent variables $\mathbf{z}_t$ and $\hat{\mathbf{z}}_{t+1}$ that serve as information bottlenecks between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, such that

$$\begin{aligned} &p(\mathbf{x}_{t+1}, \mathbf{z}_t, \hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \\ &\quad = p(\mathbf{z}_t|\mathbf{x}_t) p(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \mathbf{u}_t) p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1}). \end{aligned} \tag{9}$$

Intuitively, it is natural to interpret $\mathbf{z}_t$ and $\hat{\mathbf{z}}_{t+1}$ to be stochastic embeddings of $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, respectively.

Next, we enforce global linearity of $p(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \mathbf{u}_t)$ by restricting it to be a deterministic, linear transition function of the form

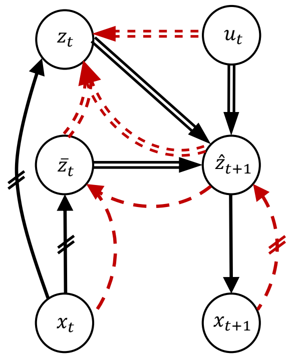

Figure 1: RCE graphical model. Black arrows show the generative links and dashed red arrows show the recognition model. Parallel lines mean deterministic links, while single lines mean stochastic links (a link that involves in sampling). $\mathbf{z}_t$ and $\bar{\mathbf{z}}_t$ are two samples from $p(\mathbf{z}|\mathbf{x})$. We use a single network (the encoder network) to model the conditional probability of the links with the hatch marks.

$$\hat{\mathbf{z}}_{t+1} = \mathbf{A}\mathbf{z}_t + \mathbf{B}\mathbf{u}_t + \mathbf{c}, \tag{10}$$

where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{c}$ are matrices that respectively define the state dynamics, control dynamics, and the offset. To emphasize the deterministic nature of this transition, we replace all the subsequent mentions of deterministic $p(\cdot|\cdot)$ transitions with $\delta(\cdot|\cdot)$.

In order to learn more expressive transition dynamics, we relax the global linearity constraint to a local one. Unlike global linearity, local linearity requires a linearization point. To account for this, we introduce an additional variable $\bar{\mathbf{z}}_t$ to serve as the linearization point, which results in a new generative model (see the black arrows in Fig. 1),

$$\begin{aligned} &p(\mathbf{x}_{t+1}, \mathbf{z}_t, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \\ &\quad = p(\mathbf{z}_t|\mathbf{x}_t) p(\bar{\mathbf{z}}_t|\mathbf{x}_t) \\ &\qquad \delta(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \bar{\mathbf{z}}_t, \mathbf{u}_t) p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1}), \end{aligned} \tag{11}$$

whose corresponding deterministic transition function for $\delta(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \bar{\mathbf{z}}_t, \mathbf{u}_t)$ is

$$\hat{\mathbf{z}}_{t+1} = \mathbf{A}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t)\mathbf{z}_t + \mathbf{B}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t)\mathbf{u}_t + \mathbf{c}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t). \tag{12}$$

Here, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{c}$ are functions of $(\bar{\mathbf{z}}_t, \mathbf{u}_t)$, and can be parameterized by neural networks. Since the linearization point $\bar{\mathbf{z}}_t$ is not known in advance, we treat $\bar{\mathbf{z}}_t$ as a random variable with distribution $p(\bar{\mathbf{z}}_t|\mathbf{x}_t)$. A natural consideration for $p(\bar{\mathbf{z}}_t|\mathbf{x}_t)$ is to set it to be identical to $p(\mathbf{z}_t|\mathbf{x}_t)$ *a priori*. This has the effect of making the iLQR controller robust to stochastic sampling of z_t during planning.

3.2 Deep Variational Learning

Training the generative model in Eq. 11 using maximum likelihood is intractable, since it requires the marginalization of the latent variables. Therefore, we propose to use deep variational inference [5, 10] and maximize the variational lower bound of the log-likelihood, instead. The variational lower bound requires us to define a variational approximation to the true posterior

$$q(\mathbf{z}, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1} | \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t) \approx p(\mathbf{z}, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1} | \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t).$$

In adherence to the interpretation of $\mathbf{z}_t$ and $\hat{\mathbf{z}}_{t+1}$ as stochastic embeddings of $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, it is important to enforce consistency between $p(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t)$ and the next step probability of the embedding given the observation $p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})$. Since we do not have access to $p(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t)$, we instead encourage this consistency through posterior regularization by explicitly setting

$$q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) = p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}). \tag{13}$$

Next, we propose a novel factorization of the full variational posterior as

$$\begin{aligned} &q(\mathbf{z}, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1} | \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t) \\ &\quad = q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1}) \delta(\mathbf{z}_t|\hat{\mathbf{z}}_{t+1}, \bar{\mathbf{z}}_t, \mathbf{u}_t), \end{aligned} \tag{14}$$

where $q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})$ is the *backward encoder* and $\delta(\mathbf{z}_t|\hat{\mathbf{z}}_{t+1}, \bar{\mathbf{z}}_t, \mathbf{u}_t)$ is the deterministic *reverse transition*. Our choice of factorization results in a recognition model that contrasts sharply with that in E2C. First, our recognition model properly conditions the inference of $\hat{\mathbf{z}}_{t+1}$ on the observation $\mathbf{x}_{t+1}$. Second, our recognition model explicitly accounts for the deterministic transition in the generative model; inference of the deterministic transition can be performed in closed-form using a deterministic reverse transition that recovers $\mathbf{z}_t$ as a function of $\bar{\mathbf{z}}_t, \mathbf{u}_t$ and $\hat{\mathbf{z}}_{t+1}$. To be consistent with Eq. 12, we require that

$$\mathbf{z}_t = \mathbf{A}_t^{-1}(\bar{\mathbf{z}}_t, \mathbf{u}_t)\big(\hat{\mathbf{z}}_{t+1} - \mathbf{B}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t)\mathbf{u}_t - \mathbf{c}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t)\big). \tag{15}$$

During the training of the generative model, we only need to access the inverse of $\mathbf{A}_t$. As such, we propose to directly train a network that outputs its inverse $\mathbf{M}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t) = \mathbf{A}_t^{-1}(\bar{\mathbf{z}}_t, \mathbf{u}_t)$. To make sure that $\mathbf{M}_t$ is an invertible matrix and to enable efficient estimation, we restrict $\mathbf{M}_t$ to be a rank-one perturbation of the identity matrix, i.e.,

$$\mathbf{M}_t = \mathbf{I}_{n_z} + \mathbf{w}_t(\bar{\mathbf{z}}_t, \mathbf{u}_t)\mathbf{r}_t^\top(\bar{\mathbf{z}}_t, \mathbf{u}_t), \tag{16}$$

where $\mathbf{I}_{n_z}$ is the identity matrix of size n_z, and $\mathbf{w}_t$ and $\mathbf{r}_t$ are two column vectors in $\mathbb{R}^{n_z}$. We constraint these vectors to be non-negative using a non-negative activation at their corresponding output layers.

We now formally the give the RCE loss and its lower bound property.

Lemma 1. *Let $\mathcal{L}_t^{RCE}$ be defined as*

$$\begin{aligned} \mathcal{L}_t^{RCE} &= \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\log p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})\big] \\ &- \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[KL\big(q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t) \parallel p(\bar{\mathbf{z}}_t|\mathbf{x}_t)\big)\big] \\ &+ H\big(q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})\big) + \mathbb{E}_{\substack{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) \\ q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})}}\big[\log p(\mathbf{z}_t|\mathbf{x}_t)\big]. \end{aligned} \tag{17}$$

Subject to the constraints we explicitly impose on q, $\mathcal{L}_t^{RCE}$ is a lower bound of the conditional log-likelihood $\log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$.

Proof. See Appendix A. □

Figure 1 contains a graphical representation of our model. We report a complete derivation of Eq. 17 in Appendix A. It is important to note that unlike the E2C encoder (Eq. 8 in [17]), our recognition model takes the future state $\mathbf{x}_{t+1}$ as input. In the case of noisy dynamics, the future state heavily influences the posterior. Thus, E2C's failure to incorporate the future state into the variational approximation of the posterior could be detrimental to the performance of the system in the noisy regime. We clearly demonstrate this phenomenon in our experiments.

3.3 Network Structure

For the four problems used in our experiments in Section 4, we use feedforward networks for encoding, decoding, and transition. Depending on the input image size, the encoder and decoder can have fully-connected layers or convolutional layers. The transition networks always have fully-connected layers. According to Eq. 17, we need to model four different conditional probabilities: $p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})$, $q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})$, $q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t)$, and $p(\mathbf{z}_t|\mathbf{x}_t)$. Figure 2 shows the high-level depiction of the networks and the connection between different variables used in these probabilities.

3.4 Planning

We use the iLQR algorithm to plan in the latent space $\mathcal{Z}$. A good latent space representation should allow us not only to reconstruct and predict the images accurately, but also to plan well in this space using $f_{\mathcal{Z}}$.

The inputs to the planning algorithm are the two high-dimensional observations $\mathbf{x}^i$ and $\mathbf{x}^f$, corresponding to the initial and final (goal) states $\mathbf{s}^i$ and $\mathbf{s}^f$. We encode these two high-dimensional observations to the latent

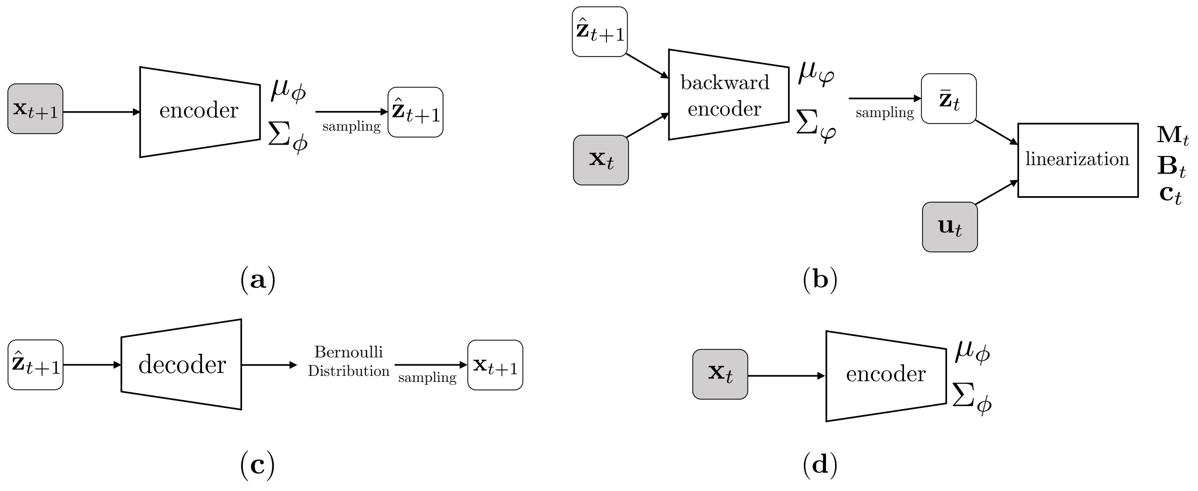

Figure 2: Schematic of the networks that are used for modeling the probabilities in our model. The gray boxes contain input (observable) variables. *(a)* Encoder network that models $q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) = \mathcal{N}\big(\mu_\phi(\mathbf{x}_{t+1}), \mathbf{\Sigma}_\phi(\mathbf{x}_{t+1})\big)$. *(b)* Transition network that contains two parts. One part, denoted by "backward encoder", models $q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1}) = \mathcal{N}\big(\mu_\varphi(\mathbf{x}_t, \hat{\mathbf{z}}_{t+1}), \mathbf{\Sigma}_\varphi(\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})\big)$, and the other part, denoted by "linearization", is used to obtain $\mathbf{M}_t$, $\mathbf{B}_t$, and $\mathbf{c}_t$, which are the parameters of the locally linear model in the latent space. *(c)* Decoder network that models $p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})$. In our experiments we assume that this distribution is Bernoulli. Therefore, we use sigmoid nonlinearity at the last layer of the decoder. $\bar{\mathbf{x}}_{t+1}$ is the reconstructed version of $\mathbf{x}_{t+1}$. *(d)* The network that models $p(\mathbf{z}_t|\mathbf{x}_t)$. According to Eq. 13, since $p(\mathbf{z}_t|\mathbf{x}_t) = q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ and therefore we tie the parameters of this network with the encoder network, $p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mu_\phi(\mathbf{x}_t), \mathbf{\Sigma}_\phi(\mathbf{x}_t))$. Note that $p(\mathbf{z}_t|\mathbf{x}_t)$ is the same as $p(\bar{\mathbf{z}}_t|\mathbf{x}_t)$. Thus, the KL term in (17) can be written as $\mathrm{KL}\big(\mathcal{N}(\mu_\varphi, \mathbf{\Sigma}_\varphi) \parallel \mathcal{N}(\mu_\phi(\mathbf{x}_t), \mathbf{\Sigma}_\phi(\mathbf{x}_t))\big)$.

space observations $\mathbf{z}^i$ and $\mathbf{z}^f$. We sample a random set of H actions $\mathbf{u}_{1:H}$ and apply them to the system, starting from the initial state $\mathbf{s}^i$ (represented in the latent space by $\mathbf{z}^i$). This generates a reference trajectory $\{\bar{\mathbf{z}}_1 = \mathbf{z}^i, \bar{\mathbf{u}}_1 = \mathbf{u}_1, \bar{\mathbf{z}}_2, \bar{\mathbf{u}}_2 = \mathbf{u}_2, \ldots, \bar{\mathbf{z}}_H, \bar{\mathbf{u}}_H = \mathbf{u}_H, \bar{\mathbf{z}}_{H+1}\}$ of size H. We pass this reference trajectory to iLQR and it returns the set of actions $\mathbf{u}^*_{1:H}$ that has been iteratively optimized to minimize a quadratic cost similar to (2) in the latent space $\mathcal{Z}$. We apply $\mathbf{u}^*_1$ to the dynamical system, observe the next state's observation $\mathbf{x}_2$, and encode it to the latent space observation $\mathbf{z}_2$. We then generate another reference trajectory by starting from $\mathbf{z}_2$ and applying the sequence of H actions $\{\mathbf{u}^*_2, \ldots, \mathbf{u}^*_H, \mathbf{u}_{H+1}\}$, where $\mathbf{u}_{H+1}$ is a random action. We then run iLQR with this trajectory and apply the first action in the set of H actions it returns to the system. We continue this process for T (the planning horizon) steps.

4 Experiments

In this section, we compare the performance of our proposed RCE model with that of E2C in terms of both prediction and planning in the four domains of [17]. To generate our training and test sets, each consists of triples $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$, we first sample a state $\mathbf{s}_t$ and generate its corresponding observation $\mathbf{x}_t$. We then take an action $\mathbf{u}_t$ and add a Gaussian noise with covariance $\mathbf{\Sigma}_{\mathbf{n}^\mathcal{S}}$ according to Eq. 1 to obtain the next state $\mathbf{s}_{t+1}$, which is used to generate the next observation $\mathbf{x}_{t+1}$. We consider both deterministic ($\mathbf{\Sigma}_{\mathbf{n}^\mathcal{S}} = \mathbf{0}$) and stochastic scenarios. In the stochastic case, we add noise to the system with different values of $\mathbf{\Sigma}_{\mathbf{n}^\mathcal{S}}$ and evaluate the models performance under noise.

In each of the four domains used in our experiments, we compare the performance of RCE and that of E2C in terms of four different factors (see Tables 1– 4). **1)** *Reconstruction Loss* is the loss in reconstructing $\mathbf{x}_t$ using the encoder and decoder. **2)** *Prediction Loss* is the loss in predicting $\mathbf{x}_{t+1}$, given $\mathbf{x}_t$ and $\mathbf{u}_t$, using the encoder, decoder, and transition network. **3)** *Planning Loss* is computed based on the following quadratic loss:

$$J = \sum_{t=1}^{T} (\mathbf{s}_t - \mathbf{s}^f)^\top \mathbf{Q} (\mathbf{s}_t - \mathbf{s}^f) + \mathbf{u}_t^\top \mathbf{R} \mathbf{u}_t. \tag{18}$$

We apply the sequence of actions returned by iLQR to the dynamical system and report the value of the loss in Eq. 18. **4)** *Success Rate* shows the number of times the agents reaches the goal within the planning horizon T, and remains near the goal in case it reaches it in less than T steps. For each of the domains, all the results are averaged over 20 runs. The details of our implementations, including the network's structure, the size of the latent space, and the planning horizon are specified in Appendix B.

4.1 Planar System

Consider an agent in a surrounded area, whose goal is to navigate from a corner to the opposite one, while avoiding the six obstacles in this area. The system is observed through a set of 40 × 40 pixel images taken from the top, which specify the agent's location in the area. Actions are two-dimensional and specify the direction of the agent's movement.

Table 1 shows that RCE outperforms E2C in both prediction/reconstruction and planning in this domain. The Gaussian noise we add to the system has a diagonal covariance matrix with equal variance in all dimensions. The values mentioned in the table for $\mathbf{\Sigma_n}s$ are the standard deviation in each dimension.

Figure 3 shows the latent space representation of data points in the planar system dataset for both RCE and E2C models. RCE has clearly a more robust representation against the noise and is able to predict the defined trajectory with a much higher quality.

4.2 Inverted Pendulum (Acrobat)

This is the classic problem of controlling an inverted pendulum [16] from 48 × 48 pixel images. The goal in this task is to swing up and balance an underactuated pendulum from a resting position (pendulum hanging down). The true state space of the system $\mathcal{S}$ has two dimensions: angle and angular velocity. To keep the Markovian property in the observation space, we need to have two images in each observation $\mathbf{x}_t$, since each image shows only position of the pendulum and does not have any information about its velocity.

Table 2 contains our results of comparing RCE and

Table 1: RCE and E2C Comparison – Planar System

$\mathbf{\Sigma_n}s$	Algorithm	Reconstruction Loss	Prediction Loss	Planning Loss	Success Rate
0	**RCE**	3.6 ± 1.7	6.2 ± 2.8	21.4 ± 2.9	100%
	E2C	7.4 ± 1.7	9.3 ± 2.8	26.3 ± 4.9	100%
1	**RCE**	8.3 ± 5.5	10.1 ± 6.2	25.4 ± 3.6	100%
	E2C	19.2 ± 5.1	28.3 ± 10.2	34.1 ± 9.5	95%
2	**RCE**	12.3 ± 4.9	17.3 ± 6.2	36.4 ± 10.3	95%
	E2C	37.1 ± 10.5	45.8 ± 13.1	63.7 ± 16.3	75%
5	**RCE**	25.2 ± 6.1	27.3 ± 8.2	50.3 ± 14.5	85%
	E2C	60.3 ± 18.2	78.3 ± 15.0	112.4 ± 30.2	45%

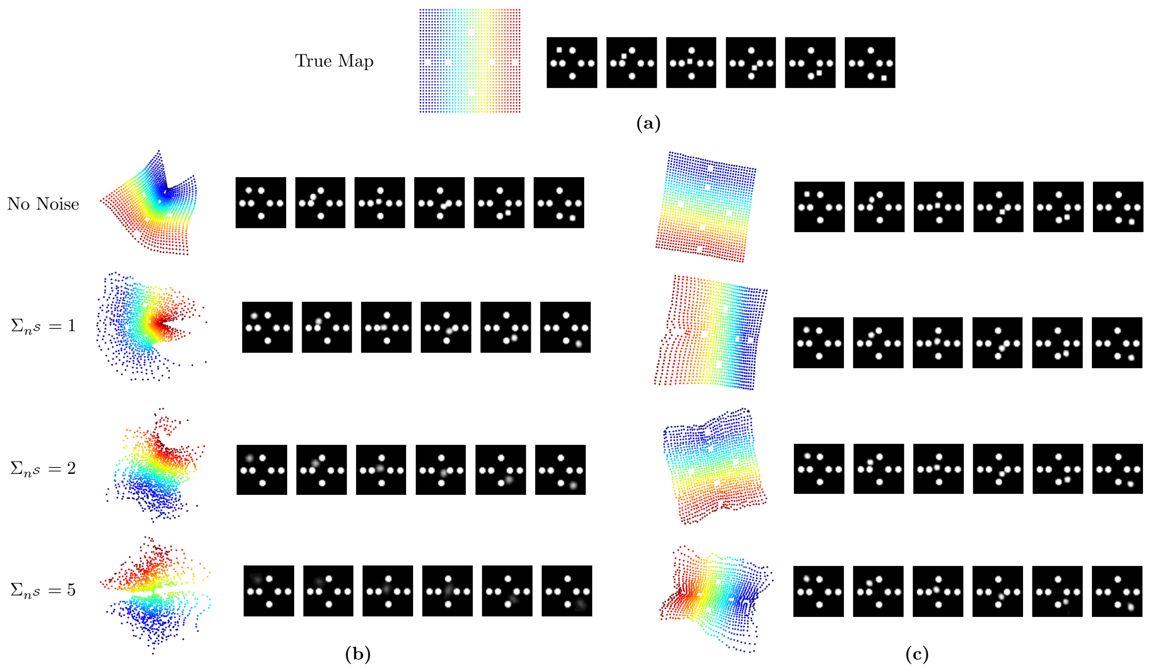

Figure 3: *(a) Left:* The true state space of the planar system. Each point on the map corresponds to one image in the dataset. *(a) Right:* A random trajectory. Each image is 40 × 40 black and white. The circles show the obstacles and the square is the agent in the domain. *(b)* Reconstructed map and predicted trajectory in the latent space of the E2C model for different noise levels. *(c)* Reconstructed map and predicted trajectory in the latent space of the RCE model for different noise levels.

Table 2: RCE and E2C Comparison – Inverted Pendulum (Acrobat)

$\Sigma_{\mathbf{n}}s$	Algorithm	Reconstruction Loss	Prediction Loss	Planning Loss	Success Rate
0	**RCE**	43.1 ± 13.2	48.1 ± 17.2	14.2 ± 4.6	95%
	E2C	73.2 ± 20.1	79.6 ± 32.6	16.1 ± 2.9	90%
1	**RCE**	61.1 ± 16.2	70.2 ± 36.1	17.3 ± 7.1	85%
	E2C	97.1 ± 34.1	109.7 ± 58.2	29.9 ± 9.2	60%
2	**RCE**	92.11 ± 35.4	106.4 ± 53.2	27.5 ± 6.6	70%
	E2C	140.2 ± 47.1	179.5 ± 61.1	40.7 ± 11.8	40%

E2C models in this task. Learning the dynamics in this problem is harder than reconstructing the images. Therefore, at the beginning of the training we set the weights of the two middle terms in Eq. 17 to 10, and eventually decrease them to 1. The results show that RCE outperforms than E2C, and the difference is significant under noisy conditions.

4.3 Cart-pole Balancing

This is the visual version of the classic task of controlling a cart-pole system [12]. The goal in this task is to balance a pole on a moving cart, while the cart avoids hitting the left and right boundaries. The control (action) is 1-dimensional and is the force applied to the cart. The original state of the system $\mathbf{s}_t$ is 4-dimensional. The observation $\mathbf{x}_t$ is a history of two 80×80 pixel images (to maintain the Markovian property). Due to the relatively large size of the images, we use convolutional layers in encoder and decoder. To make a fair comparison with E2C, we also set the dimension of the latent space $\mathcal{Z}$ to 8.

Table 3 contains our results of comparing RCE and E2C models in this task. We again observe a similar trend: RCE outperforms E2C in both noiseless and noisy settings and is significantly more robust.

4.4 Three-link Robot Arm

The goal in this task is to move a three-link planar robot arm from an initial position to a final position (both chosen randomly). The real state of the system $\mathcal{S}$ is 6-dimensional and the actions are 3-dimensional, representing the force applied to each joint of the arm. We use two 128×128 pixel images of the arm as observation $\mathbf{x}$. To be consistent with the E2C model, we choose the latent space $\mathcal{Z}$ to be 8-dimensional.

Table 4 contains our results of comparing RCE and E2C models in this task. Similar to the other domains, our results show that the RCE model is more robust to noise than E2C.

5 Conclusions

In summary, we proposed a new method to embed the high-dimensional observations of a MDP in such a way that both the embeddings and locally optimal controllers are robust w.r.t. the noise in the system's dynamics. Our RCE model enjoys a clean separation between the generative graphical model and its recognition model. The RCE's generative model explicitly treats the unknown linearization points as random variables, while the recognition model is factorized in reverse direction to take into account the future observation as well as exploiting determinism in the transition dynamics. Our experimental results demonstrate that the RCE's predictive and planning performance are better and significantly more robust than that of E2C in all the four benchmarks where E2C performance has been measured [17].

Table 3: RCE and E2C Comparison – Cart-pole Balancing

$\Sigma_{\mathbf{n}}s$	Algorithm	Reconstruction Loss	Prediction Loss	Planning Loss	Success Rate
0	**RCE**	33.2 ± 15.6	42.1 ± 26.9	21.2 ± 6.3	90%
	E2C	44.9 ± 17.0	57.3 ± 22.9	25.3 ± 4.8	85%
1	**RCE**	52.1 ± 20.3	63.3 ± 27.2	28.4 ± 5.5	80%
	E2C	70.2 ± 23.7	90.5 ± 42.4	39.8 ± 5.2	70%
2	**RCE**	77.6 ± 30.2	88.4 ± 38.3	42.2 ± 8.3	70%
	E2C	112.6 ± 39.2	133.0 ± 56.5	67.2 ± 9.3	40%

Table 4: RCE and E2C Comparison – Robot Arm

$\Sigma_{\mathbf{n}}s$	Algorithm	Reconstruction Loss	Prediction Loss	Planning Loss	Success Rate
0	**RCE**	60.5 ± 27.1	69.9 ± 32.2	81.3 ± 35.5	90%
	E2C	71.3 ± 19.5	83.4 ± 28.6	90.23 ± 47.38	90%
1	**RCE**	96.5 ± 34.4	112.6 ± 42.2	106.2 ± 50.8	80%
	E2C	138.1 ± 42.5	172.2 ± 58.3	155.2 ± 70.1	65%

References

[1] C. Atkeson and J. Murimoto. Non-parametric representations of policies and value functions: A trajectory-based approach. In *Advances in Neural Information Processing Systems*, 2002.

[2] W. Böhmer, J. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *Künstliche Intelligenz*, 29(4):353–362, 2015.

[3] D. Jacobson and D. Mayne. *Differential Dynamic Programming*. American Elsevier, 1970.

[4] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *Proceedings of ICLR*, 2017.

[5] D. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of ICLR*, 2014.

[6] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2010.

[7] S. Levine and V. Koltun. Variational policy search via trajectory optimization. In *Advances in Neural Information Processing Systems*, 2013.

[8] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of ICINCO*, pages 222–229, 2004.

[9] Y. Pan and E. Theodorou. Probabilistic differential dynamic programming. In *Advances in Neural Information Processing Systems*, 2014.

[10] D. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

[11] R. Shu, H. Bui, and M. Ghavamzadeh. Bottleneck conditional density estimation. In *Proceedings of the International Conference on Machine Learning*, 2017.

[12] R. Sutton and A. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[13] Y. Tassa, T. Erez, and W. Smart. Receding horizon differential dynamic programming. In *Advances in Neural Information Processing Systems*, 2008.

[14] E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained non-linear stochastic systems. In *Proceedings of the American Control Conference*, 2005.

[15] N. Wahlström, T. Schön, and M. Desienroth. From pixels to torques: Policy learning with deep dynamical models. In *arXiv preprint arXiv:1502.02251*, 2015.

[16] H. Wang, K. Tanaka, and M. Griffin. An approach to fuzzy control of nonlinear systems; stability and design issues. *IEEE Transactions on Fuzzy Systems*, 4(1), 1996.

[17] 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.

A Objective Function

Proof of Lemma 1. Suppose $q^\star = q(\mathbf{z}_t, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{x}_{t+1})$. Consider the factorization of $q^\star$ based on Eq. 15 and also the factorization of $p(\mathbf{x}_{t+1}, \mathbf{z}_t, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ based on Eq. 11. The variational lower bound on the conditional probability distribution $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ can be derived as following:

$$\begin{aligned}\log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) &\geq \mathbb{E}_{q^\star}\big[\log p(\mathbf{x}_{t+1}, \mathbf{z}_t, \bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) - \log q^\star)\big] \\ &= \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\log p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})\big] - \mathbb{E}_{q^\star}\big[\log q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) + \log q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t) \\ &\quad + \log \delta(\mathbf{z}_t|\bar{\mathbf{z}}_t, \hat{\mathbf{z}}_{t+1}, \mathbf{u}_t) - \log p(\mathbf{z}_t|\mathbf{x}_t) - \log p(\bar{\mathbf{z}}_t|\mathbf{x}_t) - \log \delta(\hat{\mathbf{z}}_{t+1}|\mathbf{z}_t, \bar{\mathbf{z}}_t, \mathbf{u}_t)\big]\end{aligned} \tag{19}$$

We can simply ignore the $\delta(.|.)$ terms, because the cross entropy for these terms are zero. Therefore the lower bound can be written as:

$$\begin{aligned}\log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) &\geq \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\log p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})\big] - \mathbb{E}_{\substack{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) \\ q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})}}\big[\log \frac{q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t)}{p(\bar{\mathbf{z}}_t|\mathbf{x}_t)}\big] \\ &\quad - \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\log q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})\big] - \mathbb{E}_{\substack{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) \\ q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})}}\big[\log p(\mathbf{z}_t|\mathbf{x}_t)\big] \\ &= \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\log p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})\big] - \mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\mathrm{KL}\big(q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t) \parallel p(\bar{\mathbf{z}}_t|\mathbf{x}_t)\big)\big] \\ &\quad + \mathrm{H}\big(q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})\big) + \mathbb{E}_{\substack{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) \\ q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})}}\big[\log p(\mathbf{z}_t|\mathbf{x}_t)\big] = \mathcal{L}_t^{RCE}\end{aligned} \tag{20}$$

□

The terms in $\mathcal{L}_t^{RCE}$ can be written in closed forms:

1. $\mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\big[\log p(\mathbf{x}_{t+1}|\hat{\mathbf{z}}_{t+1})\big]$
   Using the reparameterization trick [5], we should first sample from $\mathcal{N}(\mu_\phi(\mathbf{x}_{t+1}), \mathbf{\Sigma}_\phi(\mathbf{x}_{t+1}))$. Considering a Bernoulli distribution for the posterior of $\mathbf{x}_{t+1}$, the term inside the expectation is a binary cross entropy.

2. $\mathbb{E}_{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})}\Big[\mathrm{KL}\big(q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t) \parallel p(\bar{\mathbf{z}}_t|\mathbf{x}_t)\big)\Big]$
   Again, we first need to sample from $\mathcal{N}(\mu_\phi(\mathbf{x}_{t+1}), \mathbf{\Sigma}_\phi(\mathbf{x}_{t+1}))$. Note that $p(\bar{\mathbf{z}}_t|\mathbf{x}_t) = p(\mathbf{z}_t|\mathbf{x}_t)$ and $p(\mathbf{z}_t|\mathbf{x}_t) = q(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mu_\phi(\mathbf{x}_t), \mathbf{\Sigma}_\phi(\mathbf{x}_t))$. For the q_φ network, which is the transition network in our model, we have $q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t) = \mathcal{N}(\mu_\varphi, \mathbf{\Sigma}_\varphi)$. The KL term can be written as

$$\mathrm{KL}\big(q_\varphi(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_{t+1}, \mathbf{x}_t) \parallel p(\bar{\mathbf{z}}_t|\mathbf{x}_t)\big) = \frac{1}{2}\big(\mathrm{Tr}(\mathbf{\Sigma}_\phi(\mathbf{x}_t)^{-1}\mathbf{\Sigma}_\varphi) + (\mu_\phi(\mathbf{x}_t) - \mu_\varphi)^\top \mathbf{\Sigma}_\phi(\mathbf{x}_t)^{-1}(\mu_\phi(\mathbf{x}_t) - \mu_\varphi) + \log(\frac{|\mathbf{\Sigma}_\phi(\mathbf{x}_t)|}{|\mathbf{\Sigma}_\varphi|}) - n_z\big) \tag{21}$$

3. $\mathrm{H}\big(q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})\big)$
   The entropy term for the encoding network can be easily written as

$$\mathrm{H}\big(q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1})\big) = \frac{1}{2}\log\big((2\pi e)^{n_z}|\mathbf{\Sigma}_\phi(\mathbf{x}_{t+1})|\big) \tag{22}$$

4. $\mathbb{E}_{\substack{q_\phi(\hat{\mathbf{z}}_{t+1}|\mathbf{x}_{t+1}) \\ q_\varphi(\bar{\mathbf{z}}_t|\mathbf{x}_t, \hat{\mathbf{z}}_{t+1})}}\big[\log p(\mathbf{z}_t|\mathbf{x}_t)\big]$
   Here we first need to sample from $\mathcal{N}(\mu_\phi(\mathbf{x}_{t+1}), \mathbf{\Sigma}_\phi(\mathbf{x}_{t+1}))$ and $\mathcal{N}(\mu_\varphi, \mathbf{\Sigma}_\varphi)$. Given that $p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mu_\phi(\mathbf{x}_t), \mathbf{\Sigma}_\phi(\mathbf{x}_t))$, the log term inside the expectation means that we want the output of transition network to be close to the mean of its distribution, up to some constant.

$$\log p(\mathbf{z}_t|\mathbf{x}_t) = -\frac{1}{2}\Big(\log\big((2\pi e)^{n_z}|\mathbf{\Sigma}_\phi(\mathbf{x}_t)|\big) + (\mathbf{z}_t - \mu_\phi(\mathbf{x}_t))^\top \mathbf{\Sigma}_\phi(\mathbf{x}_t)^{-1}(\mathbf{z}_t - \mu_\phi(\mathbf{x}_t))\Big) \tag{23}$$

B Implementation

Transition model structure: $\mathbf{x}_t$ goes through one hidden layer with ℓ_1 units and $\hat{\mathbf{z}}_{t+1}$ goes through one hidden layer with ℓ_2 units. The outputs of the two hidden layers are concatenated and go through a network with two hidden layers of size ℓ_3 and ℓ_4, respectively, to build μ_φ and $\mathbf{\Sigma}_\varphi$. $\bar{\mathbf{z}}_t$ is sampled from this distribution and is concatenated by the action. The result goes through a three-layer network with ℓ_5, ℓ_6, and ℓ_7 units to build $\mathbf{M}_t$, $\mathbf{B}_t$, and $\mathbf{c}_t$.

In the following we will specify the values for ℓ_i's for each of the four tasks used in our experiments.

B.1 Planar system

Input: 40×40 images (1600 dimensions). 2-dimensional actions. 5000 training samples of the form $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Latent space: 2-dimensional

Encoder: 3 Layers: 300 units- 300 units- 4 units (2 for mean and 2 for the variance of the Gaussian distribution)

Decoder: 3 Layers: 300 units- 300 units- 1600 units

Transition: $\ell_1 = 100$- $\ell_2 = 5$- $\ell_3 = 100$- $\ell_4 = 4$- $\ell_5 = 20$- $\ell_6 = 20$- $\ell_7 = 10$

Number of control actions: or the planning horizon $T = 40$

B.2 Inverted Pendulum

Input: Two 48×48 images (4608 dimensions). 1-dimensional actions. 5000 training samples of the form $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Latent space: 3-dimensional

Encoder: 3 Layers: 500 units- 500 units- 6 units (3 for mean and 3 for the variance of the Gaussian distribution)

Decoder: 3 Layers: 500 units- 500 units- 4608 units

Transition: $\ell_1 = 200$- $\ell_2 = 10$- $\ell_3 = 200$- $\ell_4 = 6$- $\ell_5 = 30$- $\ell_6 = 30$- $\ell_7 = 12$

Number of control actions: or the planning horizon $T = 100$

B.3 Cart-pole Balancing

Input: Two 80×80 images (12800 dimensions). 1-dimensional actions. 15000 training samples of the form $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Latent space: 8-dimensional

Encoder: 6 Layers: convolutional layer: $32 \times 5 \times 5$; stride (1,1) - convolutional layer: $32 \times 5 \times 5$; stride (2,2) - convolutional layer: $32 \times 5 \times 5$; stride (2,2) -convolutional layer: $10 \times 5 \times 5$; stride (2,2) - 200 units- 16 units (8 for mean and 8 for the variance of the Gaussian distribution)

Decoder: 6 Layers: 200 units- 1000 units- convolutional layer: $32 \times 5 \times 5$; stride (1,1)- Upsampling (2,2)- convolutional layer: $32 \times 5 \times 5$; stride (1,1)- Upsampling (2,2)- convolutional layer: $32 \times 5 \times 5$; stride (1,1)- Upsampling (2,2)- convolutional layer: $2 \times 5 \times 5$; stride (1,1)

Transition: $\ell_1 = 300$- $\ell_2 = 10$- $\ell_3 = 300$- $\ell_4 = 16$- $\ell_5 = 40$- $\ell_6 = 40$- $\ell_7 = 32$

Number of control actions: or the planning horizon $T = 100$

B.4 Three-Link Robot Arm

Input: Two 128×128 images (32768 dimensions). 3-dimensional actions. 30000 training samples of the form $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Latent space: 8-dimensional

Encoder: 6 Layers: convolutional layer: $64 \times 5 \times 5$; stride (1,1) - convolutional layer: $32 \times 5 \times 5$; stride (2,2) - convolutional layer: $32 \times 5 \times 5$; stride (2,2) -convolutional layer: $10 \times 5 \times 5$; stride (2,2) - 500 units- 16 units (8 for mean and 8 for the variance of the Gaussian distribution)

Decoder: 6 Layers: 500 units- 2560 units- convolutional layer: $32 \times 5 \times 5$; stride (1,1)- Upsampling (2,2)- convolutional layer: $32 \times 5 \times 5$; stride (1,1)- Upsampling (2,2)- convolutional layer: $32 \times 5 \times 5$; stride (1,1)- Upsampling (2,2)- convolutional layer: $2 \times 5 \times 5$; stride (1,1)

Transition: $\ell_1 = 400$- $\ell_2 = 10$- $\ell_3 = 400$- $\ell_4 = 6$- $\ell_5 = 40$- $\ell_6 = 40$- $\ell_7 = 48$

Number of control actions: or the planning horizon $T = 100$