Latent Geodesics of Model Dynamics for Offline Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Model-based offline reinforcement learning approaches generally rely on bounds of model error. While contemporary methods achieve such bounds through an ensemble of models, we propose to estimate them using a data-driven latent metric. Particularly, we build upon recent advances in Riemannian geometry of generative models to construct a latent metric of an encoder-decoder based forward model. Our proposed metric measures both the quality of out of distribution samples as well as the discrepancy of examples in the data. We show that our metric can be viewed as a combination of two metrics, one relating to proximity and the other to epistemic uncertainty. Finally, we leverage our metric in a pessimistic model-based framework, showing a significant improvement upon contemporary model-based offline reinforcement learning benchmarks.

1 Introduction

This work focuses on leveraging Riemannian geometry of generative models in offline reinforcement learning.

Offline reinforcement learning (offline RL) (Levine et al., 2020), a.k.a. batch-mode reinforcement learning (Ernst et al., 2005; Riedmiller, 2005; Fonteneau et al., 2013), involves learning a policy from potentially suboptimal data. In contrast to imitation learning (Schaal, 1999), offline RL does not rely on expert demonstrations, but rather seeks to surpass the average performance of the agents that generated the data. Methodologies such as the gathering of new experience fall short in offline settings, requiring reassessment of fundamental learning paradigms (Buckman et al., 2020; Wang et al., 2020; Zanette, 2020).

The geometry of latent generative models has recently gained interest in unsupervised domains (Chen et al., 2018; Arvanitidis et al., 2018; Chen et al., 2020a; Arvanitidis et al., 2020). There, variational autoencoders (VAEs) have been shown to capture significant metrics in their latent representations. The resulting manifold has been shown to capture a smooth metric of the ambient output space, as well as properly capture uncertainty estimates in out of distribution (OOD) regions (Arvanitidis et al., 2018).

In this work, we introduce the aforementioned Riemannian theory of generative models to reinforcement learning. Specifically, we generalize previous results in VAEs to learn a Riemannian manifold w.r.t. the *environment's dynamics*. We achieve this by training a variational forward model of the next state. Our latent model induces a manifold and metric which capture the natural characteristics of the environment's dynamics. Moreover, we show that this metric can be analytically decoupled into metrics relating to proximity and uncertainty. Our proposed metric can be utilized in a vast range of model-based approaches in reinforcement learning (e.g., offline RL, planning). Here, we show how our learned metric can be leveraged in a model-based offline reinforcement learning framework.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Our contributions are as follows.

Technical Contributions: (1) We introduce a natural metric for forward model dynamics. The induced metric, for which we derive analytical expression for in Section 4, can be represented as a union of two metrics; namely, a metric of proximity and a metric of uncertainty. We depict the geodesics of the induced metric on a grid-like environment, suggesting our latent model captures valuable structural dependencies. (2) We integrate our metric in a model-based offline RL framework, where an agent is penalized with accordance to its distance to the data. As such, we demonstrate improved performance to contemporary offline RL approaches on several benchmarks (Section 6).

Broader Impact: Our proposed metric can be leveraged in a vast range of domains. While our work is focused on its application to offline RL, its unique characteristics can be utilized in online control, planning, and predictive analysis, as well as improve explainability of the agent and the environment. Still, using approximate models to make decisions in the real world can bring to negative social impact. Wrongful, unethical, or dangerous decisions may harm individuals affected by such actions.

2 Preliminaries

2.1 Offline Reinforcement Learning

We consider the standard Markov Decision Process (MDP) framework (Sutton et al., 1998) defined by the tuple $(\mathcal{S}, \mathcal{A}, r, P, \alpha)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ the transition kernel, and $\alpha \in (0, 1)$ is the discount factor.

In the online setting of reinforcement learning (RL), the environment initiates at some state $s_0 \sim \rho_0$. At any time step the environment is in a state $s \in \mathcal{S}$, an agent takes an action $a \in \mathcal{A}$ and receives a reward $r(s, a)$ from the environment as a result of this action. The environment transitions to state s' according to the transition function $P(\cdot|s, a)$. The goal of online RL is to find a policy $\pi(a|s)$ that maximizes the expected discounted return $v^\pi = \mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \alpha^t r(s_t, a_t)|s_0 \sim \rho_0\Big]$.

Unlike the online setting, the offline setup considers a dataset $\mathcal{D}_n = \{s_i, a_i, r_i, s'_i\}_{i=1}^n$ of transitions generated by some unknown agents. The objective of offline RL is to find the best policy in the test environment (i.e., real MDP) given only access to the data generated by the unknown agents.

2.2 Riemannian Manifolds

We define the Riemannian pullback metric, a fundamental component of our proposed method. We refer the reader to Carmo (1992) for further details on Riemannian geometry.

We are interested in studying a smooth surface M with a Riemannian metric g. A Riemannian metric is a smooth function that assigns a symmetric positive definite matrix to any point in M. At each point $z \in M$ a tangent space $T_z M$ specifies the pointing direction of vectors "along" the surface.

Definition 1. *Let M be a smooth manifold. A Riemannian metric g on M changes smoothly and defines a real scalar product on the tangent space $T_z M$ for any $z \in M$ as*

$$g_z(x, y) = \langle x, y \rangle_z = \langle x, G(z)y \rangle, \quad x, y \in T_z M,$$

where $G(z) \in \mathbb{R}^{d_z \times d_z}$ is the corresponding metric tensor. (M, g) is called a Riemannian manifold.

The Riemannian metric enables us to easily define geodesic curves. Consider some differentiable mapping $\gamma : [0, 1] \mapsto M \subseteq \mathbb{R}^{d_z}$, such that $\gamma(0) = z_0, \gamma(1) = z_1$. The length of the curve γ measured on M is given by

$$L(\gamma) = \int_0^1 \sqrt{\Big\langle \frac{\partial \gamma(t)}{\partial t}, G(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \Big\rangle} dt. \tag{1}$$

The geodesic distance $d(z_1, z_2)$ between any two points $z_1, z_2 \in M$ is then the infimum length over all curves γ for which $\gamma(0) = z_0, \gamma(1) = z_1$. That is,

$$d(z_1, z_2) = \inf_\gamma L(\gamma) \quad \text{s.t. } \gamma(0) = z_0, \gamma(1) = z_1.$$

The geodesic distance can be found by solving a system of nonlinear ordinary differential equations (ODEs) defined in the intrinsic coordinates (Carmo, 1992).

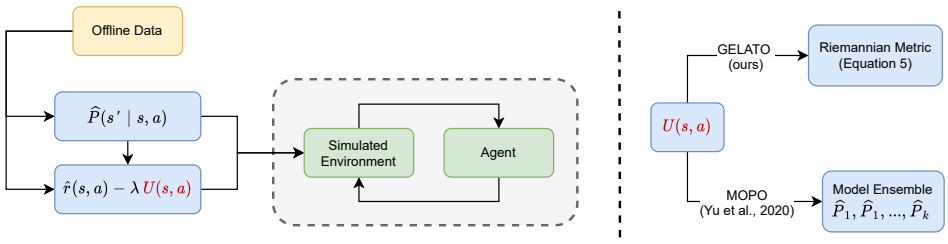


Figure 1: A reward-penalized (pessimistic) MDP is constructed from the offline data. In MOPO, the penalty is constructed using an ensemble of learned transition models. Instead, we propose to estimate the error in model dynamics through a Riemannian metric induced by a variational forward model.

Pullback Metric. Assume an ambient (observation) space $\mathcal{X}$ and its respective Riemannian manifold $(M_\mathcal{X}, g_\mathcal{X})$. Learning $g_\mathcal{X}$ can be hard (e.g., learning the distance metric between images). Still, it may be captured through a low dimensional submanifold. As such, it is many times convenient to parameterize the surface $M_\mathcal{X}$ by a latent space $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$ and a smooth function $f : \mathcal{Z} \mapsto \mathcal{X}$, where $\mathcal{Z}$ is a low dimensional latent embedding space. As learning the manifold $M_\mathcal{X}$ can be hard, we turn to learning the immersed low dimensional submanifold $M_\mathcal{Z}$ (for which the chart maps are trivial, since $\mathcal{Z} = \mathbb{R}^{d_\mathcal{Z}}$). Given a curve $\gamma : [0,1] \mapsto M_\mathcal{Z}$ we have that

$$\left\langle \frac{\partial f(\gamma(t))}{\partial t}, G_\mathcal{X}(f(\gamma(t))) \frac{\partial f(\gamma(t))}{\partial t} \right\rangle = \left\langle \frac{\partial \gamma(t)}{\partial t}, J_f^T(\gamma(t)) G_\mathcal{X}(f(\gamma(t))) J_f(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle,$$

where the Jacobian matrix $J_f(z) = \frac{\partial f}{\partial z} \in \mathbb{R}^{d_\mathcal{X} \times d_\mathcal{Z}}$ maps tangent vectors in $TM_\mathcal{Z}$ to tangent vectors in $TM_\mathcal{X}$. The induced metric is thus given by

$$G_f(z) = J_f(z)^T G_\mathcal{X}(f(z)) J_f(z). \tag{2}$$

The metric G_f is known as the *pullback metric*, as it "pulls back" the metric $G_\mathcal{X}$ on $\mathcal{X}$ back to G_f via $f : \mathcal{Z} \mapsto \mathcal{X}$. The pullback metric captures the intrinsic geometry of the immersed submanifold while taking into account the ambient space $\mathcal{X}$. The geodesic distance in ambient space is captured by geodesics in the latent space $\mathcal{Z}$, reducing the problem to learning the latent embedding space $\mathcal{Z}$ and the observation function f. Indeed, learning the latent space and observation function f can be achieved through a encoder-decoder framework, such as a VAE (Arvanitidis et al., 2018).

3 Background: Model Error in Offline RL

A key element of model-based RL methods involves estimating a model $\hat{P}(s'|s,a)$. In model-based offline RL, a pessimistic MDP[1] is constructed through an upper bound on the error of the estimated model. This work builds upon MOPO, a recently proposed model-based offline RL framework (Yu et al., 2020)). Particularly, we assume access to an approximate MDP $(\mathcal{S}, \mathcal{A}, \hat{r}, \hat{P}, \alpha)$ (e.g., trained by maximizing the likelihood of the data), and define a penalized MDP $(\mathcal{S}, \mathcal{A}, \tilde{r}, \hat{P}, \alpha)$, such that

$$\tilde{r}(s,a) = \hat{r}(s,a) - \lambda d(P(\cdot|s,a), \hat{P}(\cdot|s,a)), \forall s \in \mathcal{S}, a \in \mathcal{A},$$

where d is a given metric (e.g., the total variation distance) and $\lambda > 0$. The offline RL problem is then solved by executing an online algorithm in the reward-penalized (simulated) MDP.

Unfortunately, as $P(\cdot|s,a)$ is unknown, $d(P(\cdot|s,a), \hat{P}(\cdot|s,a))$ cannot be calculated. Nevertheless, one can attempt to upper bound the distance, i.e., for some $U : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$,

$$d(P(\cdot|s,a), \hat{P}(\cdot|s,a)) \leq U(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Figure 1 depicts the general framework. A question arises: how should $U(s,a)$ be chosen? In practice, MOPO learns an ensemble of models $\hat{P}_1, \ldots, \hat{P}_k$ to measure the upper bound $U(s,a)$. In this work, we propose to use a naturally induced metric of a variational forward model, which we show can introduce a more effective upper bound for offline RL. In Section 4 we define this metric, and finally, we leverage it to upper bound the model error in Section 5.

[1]Pessimism is a key element of offline RL algorithms (Jin et al., 2020), limiting overestimation of a trained policy due to the distribution shift between the data and the trained policy.

4 Metrics of Model Dynamics

We propose to measure the error in the model dynamics $d(P(\cdot|s,a), \hat{P}(\cdot|s,a))$ by embedding the offline data in a smooth Riemmanian manifold, equipped with a natural metric, enabling us to measure the error of out of distribution samples. Our metric is a generalization of the metric proposed by Arvanitidis et al. (2018) for generative models.

4.1 A Pullback Metric of Model Dynamics

We begin by defining the immersed Riemannian submanifold and our proposed metric. The metric is defined by a latent space $\mathcal{Z}$ and an observation function f, which will be defined later by our variational forward model.

Definition 2. *We define a Riemannian submanifold $(\mathcal{M}_{\mathcal{Z}}, g_{\mathcal{Z}})$ by a differential function $f : \mathcal{Z} \mapsto \mathcal{S}$ and latent space $\mathcal{Z}$ such that*

$$d_{\mathcal{Z}}(z_1, z_2) = \inf_{\gamma} \int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial t} \right\| dt \quad \textit{s.t. } \gamma(0) = z_1, \gamma(1) = z_2.$$

A similar metric has been used in previous work on generative latent models (Chen et al., 2018; Arvanitidis et al., 2018). It states that latent codes are close w.r.t. $d_{\mathcal{Z}}$ according to the curve which induces minimal energy in ambient observation space. It is closely related to the pullback metric (see Section 2.2), as shown by the following proposition (see Appendix for proof).

Proposition 1. *Let $(\mathcal{M}_{\mathcal{Z}}, g_{\mathcal{Z}})$ as defined above. Then $G_f(z) = J_f^T(z) J_f(z)$, for any $z \in \mathcal{Z}$.*

Indeed, Proposition 1 shows us that G_f is a pullback metric. Particularly $\mathcal{Z}$ and J_f define the structure of the ambient observation space $\mathcal{S}$. In what follows we characterize the submanifold $M_{\mathcal{Z}}$ when $z = E(s,a)$ and $f(z) \sim P(\cdot|s,a) = P(\cdot|z)$. We show that the expected pullback metric $\mathbb{E}_{P(f)}\Big[G_{\mathcal{Z}}(z)\Big]$ captures notions of proximity and uncertainty and discuss how it can be utilized to measure distance to the data manifold.

4.2 Metric of Proximity and Uncertainty of a Latent Forward Model

We consider modeling $\hat{P}(s'|s,a)$ using a generative latent model. Specifically, we consider a latent model which consists of an encoder $E : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{B}(\mathcal{Z})$ and a decoder $D : \mathcal{Z} \mapsto \mathcal{B}(\mathcal{S})$, where $\mathcal{B}(\mathcal{X})$ is set of probability measures on the Borel sets of $\mathcal{X}$. While the encoder E learns a latent representation of s, a, the decoder D estimates the next state s' according to $P(\cdot|s,a)$. This model corresponds to the decomposition $P(\cdot|s,a) = D(\cdot|E(s,a))$, where here D plays the role of the observation function f, and E maps states and actions to the latent space $\mathcal{Z}$. Such a model can be trained by maximizing the evidence lower bound (ELBO) over the data. That is, given a prior $P(z)$, we model E_ϕ, D_θ as parametric functions and maximize the ELBO

$$\max_{\theta,\phi} \mathbb{E}_{E_\phi(z|s,a)}\Big[\log D_\theta(s'|z)\Big] - D_{KL}(E_\phi(z|s,a)||P(z))$$

We refer the reader to the appendix for an exhaustive overview of training VAEs by maximum likelihood and the ELBO.

Having trained the latent model over the data $\mathcal{D}_n$, we may consider the Riemannian submanifold induced by its latent space $\mathcal{Z}$ and observation function D. Since D is stochastic, the metric $G_{\mathcal{Z}}$ also becomes stochastic, complicating analysis. Instead, Arvanitidis et al. (2018) proposed to use the expected pullback metric $\mathbb{E}[G_{\mathcal{Z}}]$, showing it is a good approximation of the underlying stochastic metric. Using Proposition 1, we have the following result (see Appendix for proof).

Theorem 1. *[Arvanitidis et al. (2018)] Assume $D(\cdot|z) \sim \mathcal{N}(\mu(z), \sigma(z)I)$. Then*

$$\mathbb{E}_{D(\cdot|z)}\Big[G_D(z)\Big] = \mathbb{E}_{D(\cdot|z)}\Big[J_D(z)^T J_D(z)\Big] = \underbrace{G_\mu(z)}_{\textit{proximity}} + \underbrace{G_\sigma(z)}_{\textit{uncertainty}}, \tag{3}$$

where $G_\mu(z) = J_\mu^T(z) J_\mu(z)$ and $G_\sigma(z) = J_\sigma^T(z) J_\sigma(z)$.

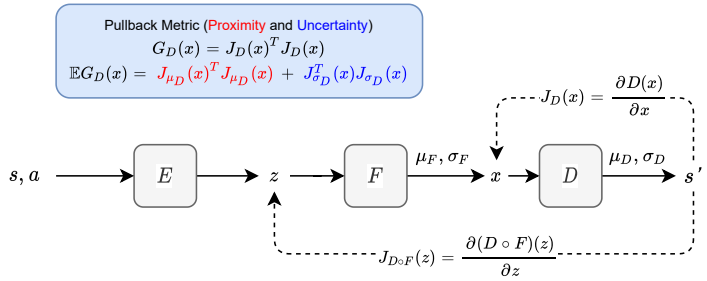


Figure 2: Plot depicts the variational latent forward model and its respective pullback metrics. Expressions for the expected pullback metrics are given in Theorems 1 and 2.

Given an embedded latent space $\mathcal{Z}$, the expected metric in Equation (3) gives us a sense of the topology of the latent space manifold induced by D. The terms $G_\mu = J_\mu^T J_\mu$ and $G_\sigma = J_\sigma^T J_\sigma$ are in fact the induced pullback metrics of μ and σ, respectively. As shortest geodesics will tend to follow small values of $\|\mathbb{E}[G_D]\|$, G_μ will keep away from areas with no latent codes, whereas G_σ will remain small in regions of low uncertainty. We therefore recognize G_μ and G_σ as *metrics of proximity and uncertainty*, respectively.

A skewed metric. Due to the inherent decoupling between proximity and uncertainty, it may be beneficial to control the curvature of the expected metric by only focusing on one of the metrics. Denoting $\alpha_{\text{prox}} \in [0, 1]$ as the proximity coefficient, we define the skewed pullback metric of D as

$$G_D^\alpha = \alpha_{\text{prox}} G_\mu + (1 - \alpha_{\text{prox}}) G_\sigma. \tag{4}$$

The skewed pullback metric will become valueable in Section 6, as we carefully control the tradeoff between proximity and uncertainty in the tested domains.

4.3 Capturing Epistemic and Aleatoric Uncertainty

The previously proposed encoder-decoder model induces a metric which captures both proximity and uncertainty w.r.t. the learned dynamics. However, the decoder variance, $\sigma(z)$, does not differentiate between aleatoric uncertainty (relating to the environment dynamics) and epistemic uncertainty (relating to missing data). To bound the validity of out of distribution (OOD) samples, we wish to capture epistemic uncertainty.

The epistemic uncertainty can be captured by methods such as model ensembles or Monte-Carlo dropout (Gal & Ghahramani, 2016). Instead, we apply an additional forward model to our previously proposed variational model. Specifically, we assume a latent model which consists of an encoder $E : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{B}(\mathcal{Z})$, forward model $F : \mathcal{Z} \mapsto \mathcal{B}(\mathcal{X})$ and decoder $D : \mathcal{X} \mapsto \mathcal{B}(\mathcal{S})$ such that $P(\cdot|s, a) = D(\cdot|x)$, and $x \sim F(\cdot|E(s, a))$. This structure enables us to capture the aleatoric uncertainty under the forward transition model F, and the epistemic uncertainty using the decoder D. That is, $\sigma_D(z)$ is used as a measure of epistemic uncertainty, as $\sigma_F(z)$ can capture the stochasticity in model dynamics. This forward model is depicted in Figure 2.

Next, we turn to analyze the pullback metric induced by the proposed forward transition model. As both F and D are stochastic (capturing epistemic and aleatoric uncertainty), the result of Theorem 1 cannot be directly applied to their composition. The following proposition provides an analytical expression for the expected pullback metric of $D \circ F$ (see Appendix for proof).

Theorem 2. *Assume* $F(\cdot|z) \sim \mathcal{N}(\mu_F(z), \sigma_F(z)I), D(\cdot|x) \sim \mathcal{N}(\mu_D(x), \sigma_D(x)I)$. *Then, the expected pullback metric of the composite function* $(D \circ F)$ *is given by*

$$\mathbb{E}_{P(D\circ F)}\Big[G_{D\circ F}(z)\Big] = J_{\mu_F}^T(z)\overline{G}_D(z)J_{\mu_F}(z) + J_{\sigma_F}^T(z)\mathit{diag}\big(\overline{G}_D(z)\big)J_{\sigma_F}(z),$$

where here, $\overline{G}_D(z) = \mathbb{E}_{x\sim F(\cdot|z)}\Big[J_{\mu_D}^T(x)J_{\mu_D}(x) + J_{\sigma_D}^T(x)J_{\sigma_D}(x)\Big]$.

Unlike the metric in Equation (3), the composite metric distorts the decoder metric with Jacobian matrices of the forward model statistics. The composite metric takes into account both proximity and uncertainty w.r.t. the ambient space as well as the latent forward model. As before, a skewed version of the metric can be designed, replacing $\overline{G}_D$ with its skewed version.

Algorithm 1 GELATO: Geometrically Enriched LATent model for Offline reinforcement learning

1: **Input:** Offline dataset $\mathcal{D}_n$, RL algorithm
2: Train variational latent forward model on dataset $\mathcal{D}_n$ by maximizing ELBO.
3: Construct approximate MDP $(\mathcal{S}, \mathcal{A}, \hat{r}, \hat{P}, \alpha)$
4: Define $\tilde{r}_d(s,a) = \hat{r}(s,a) - \lambda\Big(\frac{1}{K}\sum_{k=1}^{K} d_{\mathcal{Z}}(E(s,a), \mathrm{NN}^{(k)}_{E(s,a)})\Big)$, with distance $d_{\mathcal{Z}}$ induced by pullback metric $G_{D\circ F}$ (Theorem 2).
5: Train RL algorithm over penalized MDP $(\mathcal{S}, \mathcal{A}, \tilde{r}_d, \hat{P}, \alpha)$

5 GELATO: Incorporating the Metric in Offline RL

Having defined our metric, we are now ready to leverage it in a model based offline RL framework (see Figure 1). Specifically, provided a dataset $\mathcal{D}_n = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^n$ we train the variational latent forward model depicted in Figure 2. The model consists of an encoder E, which maps states and actions to a latent space $\mathcal{Z}$, a forward function F which maps the latent point $E(s,a)$ to a latent point $x \sim F(\cdot|E(s,a)$, and finally a decoder which maps x to the next state $s' \sim D(\cdot|x)$. The model is trained by maximizing the likelihood of state transitions in the data (a full derivation is given in the appendix). Our latent forward model induces a latent space $\mathcal{Z}$ and a pullback metric $G_{D\circ F}(z)$ which define the distance metric $d_{\mathcal{Z}}$ (Definition 2).

Algorithm 1 presents GELATO, our proposed approach. In GELATO, we estimate an upper bound, $U(s,a)$, on the model error by measuring the distance of a new sample to the data manifold. That is,

$$d(P(\cdot|s,a), \hat{P}(\cdot|s,a) \leq U(s,a) \triangleq \left(\frac{1}{K}\sum_{k=1}^{K} d_{\mathcal{Z}}(E(s,a), \mathrm{NN}^{(k)}_{E(s,a)})\right), \tag{5}$$

where here, $\mathrm{NN}^{(k)}_{E(s,a)}$ is the k^{th} nearest neighbor of $E(s,a)$ in $\mathcal{D}_n$ w.r.t. the metric d_Z. Note the sum over K nearest neighbors, allowing for more robust quantification of the distance.

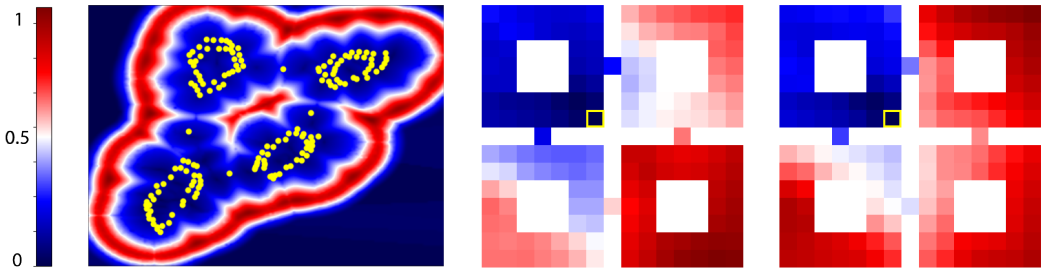
We construct the reward-penalized MDP defined in Section 3 for which the upper bound $U(s,a)$ acts as a pessimistic regularizer. Finally, we train an RL algorithm over the pessimistic MDP with transition $\hat{P}(\cdot|s,a)$ and reward $r(s,a) - \lambda U(s,a)$.

6 Experiments

6.1 Metric Visualization

To better understand the inherent structure of our metric, we constructed a grid-world environment for visualizing our proposed latent representation and metric. The 15×15 environment, as depicted in Figure 3, consists of four rooms, with impassable obstacles in their centers. The agent, residing at some position $(x, y) \in [-1, 1]^2$ in the environment can take one of four actions: up, down, left, or right – moving the agent $1, 2$ or 3 steps (uniformly distributed) in that direction. We collected a dataset of 10000 samples, taking random actions at random initializations of the environment. The ambient state space was represented by the position of the agent – a vector of dimension 2, normalized to values in $[-1, 1]$. Finally, we trained a variational latent model with latent dimension $d_{\mathcal{Z}} = 2$. We used a standard encoder $z \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$ and decoder $s' \sim \mathcal{N}(\mu_\phi(z), \sigma_\phi(z))$ represented by neural networks trained end-to-end using the evidence lower bound. We refer the reader to the appendix for an exhaustive description of the training procedure.

The latent space output of our model is depicted by yellow markers in Figure 3a. Indeed, the latent embedding consists of four distinctive clusters, structured in a similar manner as our grid-world environment. Interestingly, the distortion of the latent space accurately depicts an intuitive notion of distance between states. As such, rooms are distinctively separated, with fair distance between each cluster. States of pathways between rooms clearly separate the room clusters, forming a topology with four discernible bottlenecks.



(a) $\sqrt{\det(G_D)}$ (b) Latent Geodesic Distance (c) Latent Euclidean Distance

Figure 3: **(a)** The latent space (yellow markers) of the grid world environment and the geometric volume measure of the decoder-induced metric (background). **(b, c)** The geodesic distance of the decoder-induced submanifold and the Euclidean distance of latent states, as viewed in ambient space. All distances are calculated w.r.t. the yellow marked state. **Note**: colors in (a), which measure magnitude, are unrelated to colors in (b,c), which measure distance to the yellow marked state.

In addition to the latent embedding, Figure 3a depicts the geometric volume measure[2] $\sqrt{\det(G_D)}$ of the trained pullback metric induced by D. This quantity demonstrates the effective geodesic distances between states in the decoder-induced submanifold. Indeed geodesics between data points to points outside of the data manifold (i.e., outside of the red region), would attain high values as integrals over areas of high magnitude. In contrast, geodesics near the data manifold would attain low values.

Comparison to Euclidean distance. We visualize the decoder-induced geodesic distance and compare it to the latent Euclidean distance in Figures 3b and 3c, respectively. The plots depict the normalized distances of all states to the state marked by a yellow square. Evidently, the geodesic distance captures a better notion of distance in the said environment, correctly exposing the "land distance" in ambient space. As expected, states residing in the bottom-right room are farthest from the source state, as reaching them comprises of passing through at least two bottleneck states. In contrast, the latent Euclidean distance does not properly capture these geodesics, exhibiting a similar distribution of distances in other rooms. Nevertheless, both geodesic and Euclidean distances reveal the intrinsic topological structure of the environment, that of which is not captured by the extrinsic coordinates $(x, y) \in [-1, 1]^2$. Particularly, the state coordinates (x, y) would wrongly assign short distances to states across impassible walls or obstacles, i.e., measuring the "air distance".

6.2 Continuous Control

We performed experiments to analyze GELATO on various continuous control datasets.

Datasets. We used D4RL (Fu et al., 2020) (CC BY 4.0 license) as a benchmark for all of our experiments. We tested GELATO on three Mujoco (Todorov et al., 2012) environments (Hopper, Walker2d, Halfcheetah) on datasets generated by a single policy and a mixture of two policies. Specifically, we used datasets generated by a random agent (1M samples), a partially trained agent, i.e, medium agent (1M samples), and a mixture of partially trained and expert agents (2M samples).

Implementation Details. We trained our variational model with latent dimension $\dim(\mathcal{Z}) = 32 + \dim(\mathcal{A})$. The latent model was trained for 100k steps by stochastic gradient descent with batch size of 256. We split training into two phases. First, the model was fully trained using a calibrated Gaussian decoder (Rybkin et al., 2020). Specifically, a maximum-likelihood estimate of the variance was used $\sigma^* = \text{MSE}(\mu, \hat{\mu}) \in \arg\max_\sigma \mathcal{N}(\hat{\mu}|\mu, \sigma^2 I)$. Then, in the second stage we fit the variance decoder network.

In order to practically estimate the geodesic distance in Algorithm 1, we defined a parametric curve in latent space and used gradient descent to minimize the curve's energy. The resulting curve and pullback metric were then used to calculate the geodesic distance by a numerical estimate of the curve length (Equation (4)) (See Appendix for an exhaustive overview of the estimation method).

We used FAISS (Johnson et al., 2019) (MIT-license) for efficient GPU-based k-nearest neighbors calculation. We set $K = 20$ neighbors for the penalized reward (Equation (5)). Finally, we used a

[2]The geometric volume measure captures the volume of an infinitesimal area in the latent space.

	Hopper			Walker2d			Halfcheetah		
Method	**Rand**	**Med**	**Med-Expert**	**Rand**	**Med**	**Med-Expert**	**Rand**	**Med**	**Med-Expert**
Data Score	299	1021	1849	1	498	1062	-303	3945	8059
GELATO	**685**	**1676**	574	**412**	**1269**	**1515**	116	**5168**	6449
GELATO-unc	481	1158	879	290	487	1473	23	3034	**7130**
GELATO-prox	240	480	920	158	571	**1596**	-28	3300	**7412**
MOPO	**677**	1202	1063	**396**	518	1296	**4114**	**4974**	**7594**
MBPO	444	457	2105	251	370	222	3527	3228	907
SAC	664	325	1850	120	27	-2	3502	-839	-78
Imitation	615	1234	**3907**	47	193	329	-41	4201	4164

Table 1: Performance of GELATO and its variants in comparison to contemporary model-based methods on D4RL datasets. Scores correspond to the return, averaged over 5 seeds (standard deviation removed due to space constraints and is given in the appendix). Results for MOPO, MBPO, SAC, and imitation are taken from Yu et al. (2020). Mean score of dataset added for reference. Bold scores show an improved score w.r.t other methods.

variant of Soft Learning, as proposed by Yu et al. (2020) as our RL algorithm, trained for 1M steps. Each experiment was run on a single GPU, RTX 2080 (see Appendix for more details).

Proximity vs. Uncertainty. To test GELATO we constructed two variants, trading off proximity and uncertainty through our latent-induced metric. Particularly, we denote by GELATO-UNC and GELATO-PROX variants which implement the skewed metric (see Equation (4)), with $\alpha_{\text{prox}} = 0.1$ and $\alpha_{\text{prox}} = 0.9$, respectively. We compared GELATO and its variants to contemporary model-based offline RL approaches; namely, MOPO (Yu et al., 2020) and MBPO (Janner et al., 2019), as well as the standard baselines of SAC (Haarnoja et al., 2018) and imitation (behavioral cloning).

Results for all of the tested domains are shown in Table 1. For the non-skewed version of GELATO (i.e., $\alpha_{\text{prox}} = 0.5$) we found performance increase on most domains, and most significantly in the medium domain, i.e., partially trained agent. We believe this to be due to the inherent nature of our metric to take into account both proximity and uncertainty, allowing the agent to leverage proximity to the data in areas of high uncertainty. Since the medium dataset contained average behavior, mixing proximity benefited the agent's overall performance.

In most of the tested datasets we found an increase in performance for at least one of the GELATO variants. The med-expert datasets showed better performance for the proximity-oriented metric. These suggest flexibility of our metric to increase performance when the quality of the dataset is known, a reasonable assumption in many domains. Moreoever, the non-skewed version of GELATO, showed consistency over all datasets, favorably leveraging the strengths of proximity and uncertainty.

6.3 RBF Networks.

A question arises as to how to represent $\sigma_D(z)$. In general, neural networks may result in a poor measure of uncertainty, due to uncontrolled extrapolations of the neural network to arbitrary regions of the latent space, i.e., areas of little data. However, Arvanitidis et al. (2017) showed that the inverse variance $\beta(z) = (\sigma^2(z))^{-1}$ with a positive Radial Basis Function (RBF) network achieves a reliable uncertainty estimate, with well-behaved extrapolations in latent space. Formally the RBF network is defined by $\sigma(z) = \sqrt{(\beta(z))^{-1}}$ where $\beta(z) = W^T\phi(z), \phi_i(z) = \exp\left(-\frac{1}{2}\lambda_i \left\|z - c_i\right\|_2^2\right)$, W are the positive learned weights of the network, λ_i the bandwith, and c_i the centers trained using k-means over the learned latent representations of the offline data.

We tested GELATO on D4RL Mujoco benchmarks with an RBF decoder network. Specifically, we followed a similar training procedure with two training phases. In the first training phase we trained our variational model with a calibrated decoder as before. In the second training phase we used k-means to cluster our dataset to 128 clusters, after which an RBF network was trained for a second phase of 50000 iterations.

	Hopper		
Method	**Random**	**Medium**	**Med-Expert**
GELATO	685 ± 15	1676 ± 223	574 ± 16
GELATO-RBF	613 ± 24	1700 ± 319	498 ± 55

Table 2: Performance of GELATO using an RBF decoder compared to standard decoder.

Results for GELATO with RBF decoder networks for the Hopper environment are presented in Table 2. We did not find significant improvement in using RBF networks over decoder variance. We believe this is due to the smoothness in ambient space, allowing for well behaved extrapolations of uncertainty. We conjecture RBF networks may show improved performance on higher dimensional problems (e.g., images), yet we leave this for future work, as these may involve utilizing more involved variational models (Vahdat & Kautz, 2020).

7 Related Work

Offline Reinforcement Learning. The field of offline RL has recently received much attention as several algorithmic approaches were able to surpass standard off-policy algorithms. Value-based online algorithms do not perform well under highly off-policy batch data (Fujimoto et al., 2019; Kumar et al., 2019; Fu et al., 2019; Fedus et al., 2020; Agarwal et al., 2020), much due to issues with bootstrapping from out-of-distribution (OOD) samples. These issues become more prominent in the offline setting, as new samples cannot be acquired.

Several works on offline RL have shown improved performance on standard continuous control benchmarks (Laroche et al., 2019; Kumar et al., 2019; Fujimoto et al., 2019; Chen et al., 2020b; Swazinna et al., 2020; Kidambi et al., 2020; Yu et al., 2020; Kumar et al., 2020). In this work we were specifically interested in model-based approaches (Yu et al., 2020; Kidambi et al., 2020), in which the agent is incentivized to remain close to areas of low uncertainty. Our work focused on controlling uncertainty estimation in high dimensional environments. Our methodology utilized recent success on the geometry of deep generative models (Arvanitidis et al., 2018, 2020), proposing an alternative approach to uncertainty estimation.

Representation Learning. Representation learning seeks to find an appropriate representation of data for performing a machine-learning task (Goodfellow et al., 2016). Variational Auto Encoders (Kingma & Welling, 2013; Rezende et al., 2014) have been a popular representation learning technique, particularly in unsupervised learning regimes (Kingma et al., 2014; Sønderby et al., 2016; Chen et al., 2016; Van Den Oord et al., 2017; Hsu et al., 2017; Serban et al., 2017; Engel et al., 2017; Bojanowski et al., 2018; Ding et al., 2020), though also in supervised learning and reinforcement learning (Hausman et al., 2018; Li et al., 2019; Petangoda et al., 2019; Hafner et al., 2019, 2020). Particularly, variational models have been shown able to derive successful behaviors in high dimensional benchmarks (Hafner et al., 2020).

Various representation techniques in reinforcement learning have also proposed to disentangle representation of both states (Engel & Mannor, 2001; Littman & Sutton, 2002; Stooke et al., 2020; Zhu et al., 2020), and actions (Tennenholtz & Mannor, 2019; Chandak et al., 2019). These allow for the abstraction of states and actions to significantly decrease computation requirements, by e.g., decreasing the effective dimensionality of the action space (Tennenholtz & Mannor, 2019). Unlike previous work, GELATO is focused on measuring proximity and uncertainty for the purpose of mitigating the OOD problem and enhancing offline reinforcement learning performance.

8 Discussion and Future Work

This work presented a metric for model dynamics and its application to offline reinforcement learning. While our metric showed supportive evidence of improvement in model-based offline RL we note that it was significantly slower – comparably, 5 times slower than using the decoder's variance for uncertainty estimation. The apparent slowdown in performance was mostly due to computation of the geodesic distance. Improvement in this area may utilize techniques for efficient geodesic estimation (Chen et al., 2018, 2019).

We conclude by noting possible future applications of our work. In Section 6.1 we demonstrated the inherent geometry our model had captured, its corresponding metric, and geodesics. Still, in this work we focused specifically on metrics related to the decoded state. In fact, a similar derivation to Theorem 2 could be applied to other modeled statistics, e.g., Q-values, rewards, future actions, and more. Each distinct statistic would induce its own unique metric w.r.t. its respective probability measure. Particularly, this concept may benefit a vast array of applications in continuous or large state and action spaces.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Arvanitidis, G., Hansen, L. K., and Hauberg, S. Maximum likelihood estimation of riemannian metrics from euclidean data. In *International Conference on Geometric Science of Information*, pp. 38–46. Springer, 2017.

Arvanitidis, G., Hansen, L. K., and Hauberg, S. Latent space oddity: On the curvature of deep generative models. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Arvanitidis, G., Hauberg, S., and Schölkopf, B. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, 2020.

Bojanowski, P., Joulin, A., Lopez-Pas, D., and Szlam, A. Optimizing the latent space of generative networks. In *International Conference on Machine Learning*, pp. 600–609, 2018.

Buckman, J., Gelada, C., and Bellemare, M. G. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

Carmo, M. P. d. *Riemannian geometry*. Birkhäuser, 1992.

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., and Thomas, P. Learning action representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 941–950, 2019.

Chen, N., Klushyn, A., Kurle, R., Jiang, X., Bayer, J., and Smagt, P. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1550. PMLR, 2018.

Chen, N., Ferroni, F., Klushyn, A., Paraschos, A., Bayer, J., and van der Smagt, P. Fast approximate geodesics for deep generative models. In *International Conference on Artificial Neural Networks*, pp. 554–566. Springer, 2019.

Chen, N., Klushyn, A., Ferroni, F., Bayer, J., and van der Smagt, P. Learning flat latent manifolds with vaes. 2020a.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020b.

Ding, Z., Xu, Y., Xu, W., Parmar, G., Yang, Y., Welling, M., and Tu, Z. Guided variational autoencoder for disentanglement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7920–7929, 2020.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Engel, J., Hoffman, M., and Roberts, A. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, 2017.

Engel, Y. and Mannor, S. Learning embedded maps of markov processes. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 138–145, 2001.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.

Fonteneau, R., Murphy, S. A., Wehenkel, L., and Ernst, D. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of operations research*, 208(1):383–416, 2013.

Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*, pp. 2021–2030, 2019.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Guss, W. H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., and Salakhutdinov, R. MineRL: A large-scale dataset of Minecraft demonstrations. *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019. URL `http://minerl.io`.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Hsu, W.-N., Zhang, Y., and Glass, J. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pp. 1878–1889, 2017.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline rl? *arXiv preprint arXiv:2012.15085*, 2020.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Kalatzis, D., Eklund, D., Arvanitidis, G., and Hauberg, S. Variational autoencoders with riemannian brownian motion priors. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR, 2020.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27:3581–3589, 2014.

Klushyn, A., Chen, N., Kurle, R., Cseke, B., and van der Smagt, P. Learning hierarchical priors in vaes. In *Advances in Neural Information Processing Systems*, pp. 2870–2879, 2019.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pp. 11784–11794, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Laroche, R., Trichelair, P., and Des Combes, R. T. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pp. 3652–3661. PMLR, 2019.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, M., Wu, L., Jun, W., and Ammar, H. B. Multi-view reinforcement learning. In *Advances in neural information processing systems*, pp. 1420–1431, 2019.

Littman, M. L. and Sutton, R. S. Predictive representations of state. In *Advances in neural information processing systems*, pp. 1555–1561, 2002.

Petangoda, J. C., Pascual-Diaz, S., Adam, V., Vrancx, P., and Grau-Moya, J. Disentangled skill embeddings for reinforcement learning. *arXiv preprint arXiv:1906.09223*, 2019.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Rybkin, O., Daniilidis, K., and Levine, S. Simple and effective vae training with calibrated decoders. *arXiv preprint arXiv:2006.13202*, 2020.

Schaal, S. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6): 233–242, 1999.

Senge, R., Bösner, S., Dembczyński, K., Haasenritter, J., Hirsch, O., Donner-Banzhoff, N., and Hüllermeier, E. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255:16–29, 2014.

Serban, I., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Swazinna, P., Udluft, S., and Runkler, T. Overcoming model bias for robust offline deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

Tennenholtz, G. and Mannor, S. The natural language of actions. In *International Conference on Machine Learning*, pp. 6196–6205, 2019.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Wang, R., Foster, D. P., and Kakade, S. M. What are the statistical limits of offline rl with linear function approximation? *arXiv preprint arXiv:2010.11895*, 2020.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Zanette, A. Exponential lower bounds for batch reinforcement learning: Batch rl can be exponentially harder than online rl. *arXiv preprint arXiv:2012.08005*, 2020.

Zhu, J., Xia, Y., Wu, L., Deng, J., Zhou, W., Qin, T., and Li, H. Masked contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2010.07470*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] computational limitation (discussion section).
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] broader impact in introduction section
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] standard deviation in Table 1
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] FAISS is under MIT license. Our usage of D4RL dataset does not require consent according to CC BY 4.0 license.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] Data is of standard Mujoco benchmarks
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

	Hopper			Walker2d			Halfcheetah		
Method	**Random**	**Medium**	**Med-Expert**	**Random**	**Medium**	**Med-Expert**	**Random**	**Medium**	**Med-Expert**
Data Score	299 ± 200	1021 ± 314	1849 ± 1560	1 ± 6	498 ± 807	1062 ± 1576	-303 ± 79	3945 ± 494	8059 ± 4204
GELATO	**685 ± 15**	**1676 ± 223**	574 ± 16	**412 ± 85**	**1269 ± 549**	**1515 ± 379**	116 ± 43	**5168 ± 849**	6449 ± 2790
GELATO-unc	481 ± 29	1158 ± 423	879 ± 153	290 ± 79	487 ± 289	1473 ± 389	23 ± 35	3034 ± 585	**7130 ± 3780**
GELATO-prox	240 ± 22	480 ± 15	920 ± 249	158 ± 35	571 ± 326	**1596 ± 416**	-28 ± 31	3300 ± 613	**7412 ± 3166**
MOPO	**677 ± 13**	1202 ± 400	1063 ± 193	**396 ± 76**	518 ± 560	1296 ± 374	**4114 ± 312**	**4974 ± 200**	**7594 ± 4741**
MBPO	444 ± 193	457 ± 106	2105 ± 1113	251 ± 235	370 ± 221	222 ± 99	3527 ± 487	3228 ± 2832	907 ± 1185
SAC	664	325	1850	120	27	-2	3502	-839	-78
Imitation	615	1234	**3907**	47	193	329	-41	4201	4164

Table 3: Results of GELATO as presented in Table 1 with added std for each run, averaged over 5 seeds.

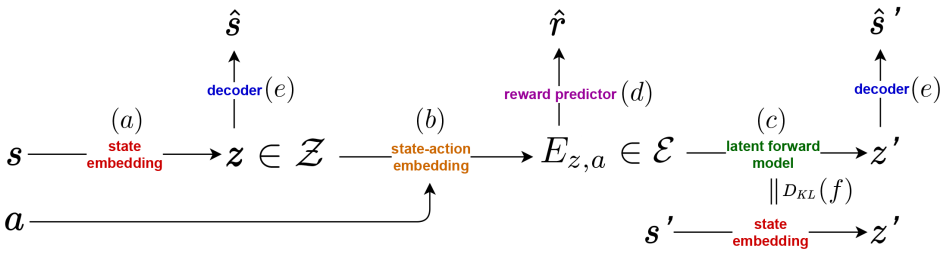


Figure 4: A graphical representation of our latent variable model. **(a)** The state s is embedded via the state embedding function (i.e., approximate posterior) $z \sim q(\cdot|s)$. **(b)** The action and embedded state pass through an invertible embedding function E to produce the state-action embedding $E_{z,a}$. **(c,d)** The state-action embedding is passed through a reward predictor and latent forward model, $\hat{r} \sim P(\cdot|E_{z,a})$ and $z' \sim P(\cdot|z,a)$, respectively. **(e)** The next latent state z' is decoded back to observation space to generate $\hat{s}' \sim P(\cdot|z')$. **(f)** Finally, during training, the target state s' is embedded and compared to z' (by the KL-divergence term in Equation (7)), preserving the consistency of the latent space $\mathcal{Z}$.

Appendix

9 Variational Latent Model

We begin by describing our variational forward model. The model, based on an encoder, latent forward function, and decoder framework assumes the underlying dynamics and reward are governed by a state-embedded latent space $\mathcal{Z} \subseteq \mathbb{R}^{d_{\mathcal{Z}}}$. The probability of a trajectory $\tau = (s_0, a_0, r_0, \ldots, s_h, a_h, r_h)$ is given by

$$P(\tau) = \int_{z_0,\ldots,z_h} P(z_0) \prod_{i=0}^{h} P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i}) \prod_{j=1}^{h} P(z_j|E_{z_{j-1},a_{j-1}}) dz_0 \ldots dz_h, \quad (6)$$

where $E : \mathcal{Z} \times \mathcal{A} \mapsto \mathcal{E} \subseteq \mathbb{R}^{d_{\mathcal{E}}}$ is a deterministic, invertible embedding function which maps pairs (z, a) to a state-action-embedded latent space $\mathcal{E}$. $E_{z,a}$ is thus a sufficient statistic of (z, a). The proposed graphical model is depicted in Figure 4. We note that an extension to the partially observable setting replaces s_t with $h_t = (s_0, a_0, \ldots, s_t)$, a sufficient statistic of the unknown state.

Maximizing the log-likelihood $\log P(\tau)$ is hard due to intractability of the integral in Equation (6). We therefore introduce the approximate posterior $q(z|s)$ and maximize the evidence lower bound.

To clear notations we define $E_{z_{-1},a_{-1}} = 0$, so that we can rewrite the above expression as

$$P(s_0, a_0, r_0, \ldots, s_h, a_h, r_h) = \int_{z_0,\ldots,z_h} \prod_{i=0}^{h} P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i})P(z_j|E_{z_{-1},a_{-1}})$$

Introducing $q(z_i|s_i)$ we can write

$$
\begin{aligned}
&\log \int_{z_0,\dots,z_h} \prod_{i=0}^{h} \frac{q(z_i|s_i)}{q(z_i|s_i)} \prod_{i=0}^{h} P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i})P(z_j|E_{z_{-1},a_{-1}}) \\
&\geq \int_{z_0,\dots,z_h} \prod_{i=0}^{h} q(z_i|s_i) \left(\sum_{i=0}^{h} \log \left(\frac{P(s_i|z_i)\pi(a_i|s_i)P(r_i|E_{z_i,a_i})P(z_j|E_{z_{-1},a_{-1}})}{q(z_i|s_i)} \right) \right) \\
&= \sum_{i=0}^{H} \mathbb{E}_{q(z_i|s_i)} \Big[\log(P(s_i \mid z_i)\pi(a_i \mid s_i)P(r_i|E_{z_i,a_i})) \Big] \\
&- \sum_{i=0}^{H-1} \mathbb{E}_{q(z_i|s_i)} \Big[D_{KL}(q(z_{i+1} \mid s_{i+1}) \mid\mid P(z_{i+1}|E_{z_i,a_i})) \Big] \\
&- D_{KL}(q(z_0 \mid s_0) \mid\mid P(z_0)).
\end{aligned}
$$

Hence,

$$
\begin{aligned}
&\sum_{i=0}^{h} \mathbb{E}_{z_i \sim q(z_i|s_i)} \Big[\log(P(s_i \mid z_i)\pi(a_i \mid s_i)P(r_i|E_{z_i,a_i})) \Big] \\
&- \sum_{i=0}^{h-1} \mathbb{E}_{z_i \sim q(z_i|s_i)} \Big[D_{KL}(q(z_{i+1} \mid s_{i+1}) \mid\mid P(z_{i+1}|E_{z_i,a_i})) \Big] \\
&- D_{KL}(q(z_0 \mid s_0) \mid\mid P(z_0)).
\end{aligned} \tag{7}
$$

The distribution parameters of the approximate posterior $q(z|s)$, the likelihoods $P(s|z), \pi(a|s), P(r|E_{z,a})$, and the latent forward model $P(z'|E_{z,a})$ are represented by neural networks. The invertible embedding function E is represented by an invertible neural network, e.g., affine coupling, commonly used for normalizing flows (Dinh et al., 2014). Though various latent distributions have been proposed (Klushyn et al., 2019; Kalatzis et al., 2020), we found Gaussian parametric distributions to suffice for all of our model's functions. Particularly, we used two outputs for every distribution, representing the expectation μ and variance σ. All networks were trained end-to-end to maximize the evidence lower bound in Equation (7).

Our latent variable model is designed to capture both the epistemic and aleatoric uncertainty (Senge et al., 2014). The variance output of the decoder captures epistemic uncertainty , while stochasticity of the latent forward model $P(z'|E_{z,a})$ captures aleatoric uncertainty. For the purpose of offline RL, we will focus on the epistemic uncertainty of our model.

We tested the quality of our variational model on datasets of two tasks in Minecraft (Guss et al., 2019); namely, a navigation task (150k examples) and a tree chopping task (250k examples), both generated by human players. The variational model was trained only on the navigation task. We

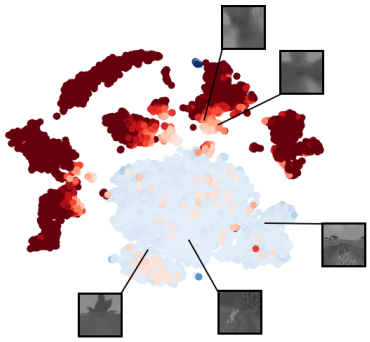


Figure 5: TSNE projection of latent space $\mathcal{Z}$ for navigation dataset (blue) and tree chopping dataset (red) in Minecraft (Guss et al., 2019). Darker colors correspond to higher decoder variance.

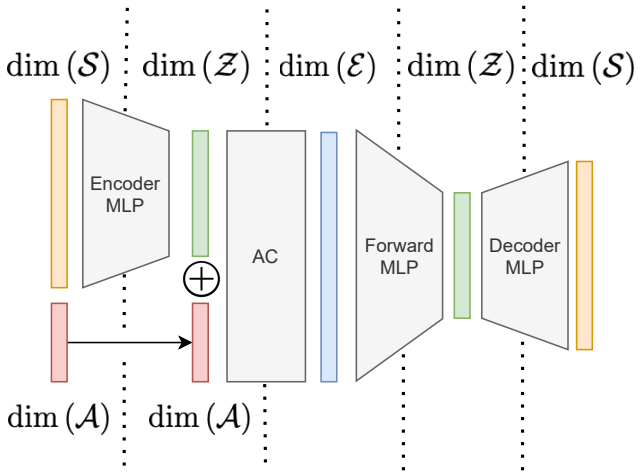


Figure 6: Latent model architecture (does not depict reward MLP).

embedded the data from both datasets using our trained model, and measured the decoder variance for all samples. Figure 5 depicts a TSNE projection of the latent space $\mathcal{Z}$, coloring in blue the navigation task and in red the tree chopping task. Light colors correspond to low variance (i.e., sharp images), whereas dark colors correspond to large variance (i.e. OOD samples). We found that our variational model was able to properly distinguish between the two tasks, with some overlap due to similarity in state space features. Additionally, we noticed a clear transition in decoding variance as samples farther away from the trained latent data attained larger variance, suggesting our variational model was properly able to distinguish OOD samples.

We refer the reader to the appendix for further analysis and approaches of uncertainty quantification in variational models. In our experiments, we found that the standard decoder variance sufficed for all of the tested domains.

10 Specific Implementation Details

As a preprocessing step rewards were normalized to values between $[-1, 1]$. We trained our variational model with latent dimensions $\dim(\mathcal{Z}) = 32$ and $\dim(\mathcal{E}) = \dim(\mathcal{Z}) + \dim(\mathcal{A})$. All domains were trained with the same hyperparameters. Specifically, we used a 2-layer Multi Layer Perceptron (MLP) to encode $\mathcal{Z}$, after which a 2-layer Affine Coupling (AC) (Dinh et al., 2014) was used to encode $\mathcal{E}$. We also used a 2-layer MLP for the forward, reward, and decoder models. All layers contained 256 hidden layers.

The latent model was trained in two separate phases for 100k and 50k steps each by stochastic gradient descent and the ADAM optimizer (Kingma & Ba, 2014). First, the model was fully trained using a calibrated Gaussian decoder (Rybkin et al., 2020). Specifically, a maximum-likelihood estimate of the variance was used $\sigma^* = \text{MSE}(\mu, \hat{\mu}) \in \arg\max_\sigma \mathcal{N}(\hat{\mu}|\mu, \sigma^2 I)$. Finally, in the second stage we fit the variance decoder network. We found this process of to greatly improve convergence speed and accuracy, and mitigate posterior collapse. We used a minimum variance of 0.01 for all of our stochastic models.

To further stabilize training we used a momentum encoder. Specifically we updated a target encoder as a slowly moving average of the weights from the learned encoder as

$$\bar{\theta} \leftarrow (1 - \tau)\bar{\theta} + \tau\theta$$

Hyperparameters for our variational model are summarized in Table 4. The latent architecture is visualized in Figure 6.

Parameter	Value	Parameter	Value
$\dim(\mathcal{Z})$	32	LEARNING RATE	10^{-3}
$\dim(\mathcal{E})$	$32 + \dim(\mathcal{A})$	BATCH SIZE	128
ENCODER MLP HIDDEN	$256, 256$	TARGET UPDATE τ	0.01
FORWARD MLP HIDDEN	$256, 256$	TARGET UPDATE INTERVAL	1
DECODER MLP HIDDEN	$256, 256$	PHASE 1 UPDATES	100000
REWARD MLP HIDDEN	$256, 256$	PHASE 2 UPDATES	50000

Table 4: Hyper parameters for variational model

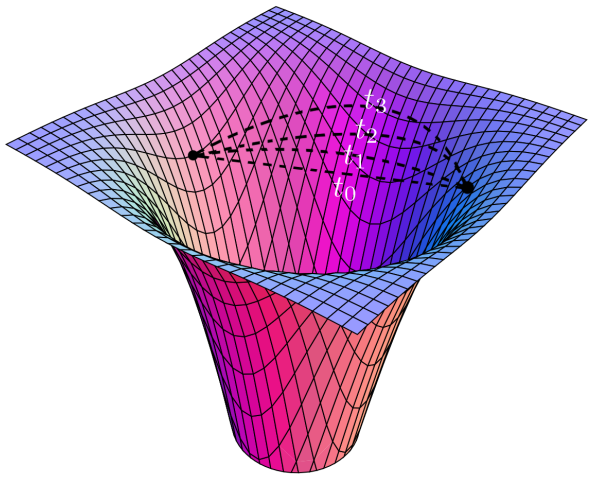


Figure 7: Illustration of geodesic curve optimization in Algorithm 2.

10.1 Geodesic Distance Estimation

In order to practically estimate the geodesic distance between two points $e_1, e_2 \in \mathcal{E}$ we defined a parametric curve in latent space and used gradient descent to minimize the curve's energy [3]. The resulting curve and pullback metric were then used to calculate the geodesic distance by a numerical estimate of the curve length (Equation (4)).

Pseudo code for Geodesic Distance Estimation is shown in Algorithm 2. Our curve was modeled as a cubic spline with 8 coefficients. We used SGD (momentum 0.99) to optimize the curve energy over 20 gradient iterations with a grid of 10 points and a learning rate of 10^{-3}. An illustration of the convergence of such a curve is illustrated in Figure 7

10.2 RL algorithm

Our learning algorithm is based on the Soft Learning framework proposed in Algorithm 2 of Yu et al. (2020). Pseudo code is shown in Algorithm 3. Specifically we used two replay buffers $\mathcal{R}_{\text{model}}, \mathcal{R}_{\text{data}}$, where $|R_{\text{model}}| = 50000$ and $\mathcal{R}_{\text{data}}$ contained the full offline dataset. In every epoch an initial state s_0 was sampled from the offline dataset and embedded using our latent model to generate $z_0 \in \mathcal{Z}$. During rollouts of π, embeddings $E_{z,a} \in \mathcal{E}$ were then generated from z and used to (1) sample next latent state z', (2) sample estimated rewards r, and (3) compute distances to $K = 20$ nearest neighbors in embedded the dataset.

We used Algorithm 2 to compute the geodesic distances, and FAISS (Johnson et al., 2019) for efficient nearest neighbor computation on GPUs. To stabilize learning, we normalized the penalty $\frac{1}{K}\sum_{k=1}^{K} d_k$ according to the maximum penalty, ensuring penalty lies in $[0, 1]$ (recall that the latent reward predictor was trained over normalized rewards in $[-1, 1]$). For the non-skewed version of

[3] Other methods for computing the geodesic distance include solving a system of ODEs (Arvanitidis et al., 2018), using graph based geodesics (Chen et al., 2019), or using neural networks (Chen et al., 2018).

Algorithm 2 Geodesic Distance Estimation

Input: forward latent F, decoder D, learning rate λ, number of iterations T, grid size n, eval points e_0, e_1
Initialize: parametric curve $\gamma_\theta : \gamma_\theta(0) = e_0, \gamma_\theta(1) = e_1$
for $t = 1$ **to** T **do**
  $L_\mu(\theta) \leftarrow \sum_{i=1}^{n} \mu_D(\mu_F(\gamma_\theta(\frac{i}{n}))) - \mu_D(\mu_F(\gamma_\theta(\frac{i-1}{n})))$
  $L_\sigma(\theta) \leftarrow \sum_{i=1}^{n} \sigma_D(\mu_F(\gamma_\theta(\frac{i}{n}))) - \sigma_D(\mu_F(\gamma_\theta(\frac{i-1}{n})))$
  $L(\theta) \leftarrow L_\mu(\theta) + L_\sigma(\theta)$
  $\theta \leftarrow \theta - \lambda \nabla_\theta L(\theta)$
end for
$G_{D\circ F} = J_{\mu_F}^T \overline{G}_D J_{\mu_F} + J_{\sigma_F}^T \text{diag}(\overline{G}_D) J_{\sigma_F}$
$\forall i, \Delta_i \leftarrow \gamma_\theta(\frac{i}{n}) - \gamma_\theta(\frac{i-1}{n})$
$d(e_0, e_1) \leftarrow \sum_{i=1}^{n} \left(\frac{\partial \gamma_\theta}{\partial t}\big|_{\frac{i}{n}}\right)^T G_{D\circ F}(\gamma_\theta(\frac{i}{n})) \left(\frac{\partial \gamma_\theta}{\partial t}\big|_{\frac{i}{n}}\right) \Delta_i$
Return: $d(e_0, e_1)$

Algorithm 3 GELATO with Soft Learning

Input: Reward penalty coefficient λ, rollout horizon h, rollout batch size b, training epochs T, number of neighbors K.
Train variational latent forward model on dataset $\mathcal{D}$ by maximizing ELBO (Equation (7))
Construct embedded dataset $\mathcal{D}_{\text{embd}} = \{E_i\}_{i=1}^{n}$ using latent model to initialize KNN.
Initialize policy π and empty replay buffer $\mathcal{R}_{\text{model}} \leftarrow \emptyset$.
for epoch $= 1$ **to** T **do**
  for $i = 1$ **to** b (in parallel) **do**
    Sample state s_1 from $\mathcal{D}$ for the initialization of the rollout and embed using latent model to produce z_1.
    for $j = 1$ **to** h **do**
      Sample an action $a_j \sim \pi(\cdot|z_j)$.
      Embed $(z_j, a_j) \rightarrow E_{z_j,a_j}$ using latent model
      Sample z_{j+1} from latent forward model $F(E_{z_j,a_j})$.
      Sample r_j from latent reward model $R(E_{z_j,a_j})$.
      Use Algorithm 2 to compute K nearest neighbors $\left\{\text{NN}_{z_j,a_j}^{(k)}\right\}_{k=1}^{K}$ and their distances $\{d_k\}_{k=1}^{N}$ to E_{z_j,a_j}.
      Compute $\tilde{r}_j = r_j - \lambda\left(\frac{1}{K}\sum_{k=1}^{K} d_k\right)$
      Add sample $(z_j, a_j, \tilde{r}_j, z_{j+1})$ to $\mathcal{R}_{\text{model}}$.
    end for
  end for
  Drawing samples from $\mathcal{R}_{\text{data}} \cup \mathcal{R}_{\text{model}}$, use SAC to update π.
end for

GELATO, we used $\lambda = 1$ as our reward penalty coefficient and $\lambda = 2$ for the skewed versions. We used rollout horizon of $h = 5$, and did not notice significant performance improvement for different values of h.

11 Missing Proofs

11.1 Proof of Proposition 1

For any curve γ, we have that

$$\begin{aligned}\int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial t} \right\| dt &= \int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial \gamma(t)}^T \frac{\partial \gamma(t)}{\partial t} \right\| dt \\ &= \int_0^1 \left\| J_f(\gamma(t))^T \frac{\partial \gamma(t)}{\partial t} \right\| dt \\ &= \int_0^1 \sqrt{\left\langle \frac{\partial \gamma(t)}{\partial t}, J_f^T(\gamma(t)) J_f(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle} dt \\ &= \int_0^1 \sqrt{\left\langle \frac{\partial \gamma(t)}{\partial t}, G_f(\gamma(t)) \frac{\partial \gamma(t)}{\partial t} \right\rangle} dt.\end{aligned}$$

This completes the proof.

11.2 Proof of Theorems 1 and 2

We begin by restating the theorems.

Theorem 1. *[Arvanitidis et al. (2018)] Assume $D(\cdot|z) \sim \mathcal{N}(\mu(z), \sigma(z)I)$. Then*

$$\mathbb{E}_{D(\cdot|z)}\Big[G_D(z)\Big] = \mathbb{E}_{D(\cdot|z)}\Big[J_D(z)^T J_D(z)\Big] = \underbrace{G_\mu(z)}_{\textit{proximity}} + \underbrace{G_\sigma(z)}_{\textit{uncertainty}}, \tag{3}$$

where $G_\mu(z) = J_\mu^T(z) J_\mu(z)$ and $G_\sigma(z) = J_\sigma^T(z) J_\sigma(z)$.

Theorem 2. *Assume $F(\cdot|z) \sim \mathcal{N}(\mu_F(z), \sigma_F(z)I), D(\cdot|x) \sim \mathcal{N}(\mu_D(x), \sigma_D(x)I)$. Then, the expected pullback metric of the composite function $(D \circ F)$ is given by*

$$\mathbb{E}_{P(D\circ F)}\Big[G_{D\circ F}(z)\Big] = J_{\mu_F}^T(z) \overline{G}_D(z) J_{\mu_F}(z) + J_{\sigma_F}^T(z) \textit{diag}\big(\overline{G}_D(z)\big) J_{\sigma_F}(z),$$

where here, $\overline{G}_D(z) = \mathbb{E}_{x\sim F(\cdot|z)}\Big[J_{\mu_D}^T(x) J_{\mu_D}(x) + J_{\sigma_D}^T(x) J_{\sigma_D}(x)\Big]$.

Notice that Theorem 1 is a special case of Theorem 2 with F being the trivial identity function. Additionally, a complete proof of Theorem 1 can be found in Arvanitidis et al. (2018). We turn to prove Theorem 2.

We begin by proving the following auxilary lemma.

Lemma 1. *Let $\epsilon \sim \mathcal{N}(0, I_K)$, $f : \mathbb{R}^d \mapsto \mathbb{R}^K$, $A \in \mathbb{R}^{K\times K}$. Denote $S_i = \textit{diag}\Big(\frac{\partial f^1}{\partial z_i}, \frac{\partial f^2}{\partial z_i}, \ldots, \frac{\partial f^K}{\partial z_i}\Big)$ for $1 \leq i \leq d$ and*

$$B = \left[S_1\epsilon, S_2\epsilon, \ldots, S_d\epsilon\right]_{K\times d}.$$

Then $\mathbb{E}\left[B^T A B\right] = J_f^T \textit{diag}(A) J_f$.

Proof. We have that

$$\begin{aligned}\mathbb{E}\left[B^T A B\right] &= \mathbb{E}\left[\begin{bmatrix}\epsilon^T S_1^T \\ \epsilon^T S_2^T \\ \vdots \\ \epsilon^T S_d^T\end{bmatrix} A \left[S_1\epsilon, S_2\epsilon, \ldots, S_d\epsilon\right]\right] \\ &= \begin{bmatrix}\mathbb{E}\left[\epsilon^T S_1^T A S_1 \epsilon\right], \mathbb{E}\left[\epsilon^T S_1^T A S_2 \epsilon\right], \ldots, \mathbb{E}\left[\epsilon^T S_1^T A S_d \epsilon\right], \\ \mathbb{E}\left[\epsilon^T S_2^T A S_1 \epsilon\right], \mathbb{E}\left[\epsilon^T S_2^T A S_2 \epsilon\right], \ldots, \mathbb{E}\left[\epsilon^T S_2^T A S_d \epsilon\right], \\ \ddots \\ \mathbb{E}\left[\epsilon^T S_d^T A S_1 \epsilon\right], \mathbb{E}\left[\epsilon^T S_d^T A S_2 \epsilon\right] \ldots, \mathbb{E}\left[\epsilon^T S_d^T A S_d \epsilon\right]\end{bmatrix}.\end{aligned}$$

Finally notice that for any matrix M

$$\mathbb{E}\left[\epsilon^T M \epsilon\right] = \sum_{i=1}^{d}\sum_{j=1}^{d} M_{ij}\mathbb{E}\left[\epsilon_i \epsilon_j\right] = \sum_{i=1}^{d} M_{ii} = \text{trace}(M).$$

Then,

$$\mathbb{E}\left[B^T A B\right] = \begin{bmatrix} \text{trace}\big(S_1^T A S_1\big), \text{trace}\big(S_1^T A S_2\big), \ldots, \text{trace}\big(S_1^T A S_d\big) \\ \text{trace}\big(S_2^T A S_1\big), \text{trace}\big(S_2^T A S_2\big), \ldots, \text{trace}\big(S_2^T A S_d\big) \\ \ddots \\ \text{trace}\big(S_d^T A S_1\big), \text{trace}\big(S_d^T A S_2\big), \ldots, \text{trace}\big(S_d^T A S_d\big) \end{bmatrix}.$$

□

Next, note that

$$\text{trace}(S_i A S_j) = \sum_{k=1}^{K} \frac{\partial f^k}{\partial z_i}\frac{\partial f^k}{\partial z_j} A_{kk}.$$

Therefore,

$$\mathbb{E}\left[B^T A B\right] = J_f^T \text{diag}(A) J_f.$$

We are now ready to prove Theorem 2.

Proof of Theorem 2. We can write $z' = \mu_F(z) + \sigma_F(z) \odot \epsilon_F$ and $s' = \mu_D(z') + \sigma_D(z') \odot \epsilon_D$ where $\epsilon_F \sim \mathcal{N}(0, I_d), \epsilon_D \sim \mathcal{N}(0, I_K)$, $\mu_F : \mathbb{R}^d \mapsto \mathbb{R}^\ell$,$\mu_D : \mathbb{R}^\ell \mapsto \mathbb{R}^K$ and $\sigma_F : \mathbb{R}^d \mapsto \mathbb{R}^\ell, \sigma_D : \mathbb{R}^\ell \mapsto \mathbb{R}^K$.

Applying the chain rule we get

$$J_{D\circ F} = \frac{\partial (D \circ F)}{\partial z} = J_{\mu_D} J_{\mu_F} + J_{\mu_D} B_{\epsilon_F} + B_{\epsilon_D} J_{\mu_F} + B_{\epsilon_D} B_{\epsilon_F}$$

where

$$\begin{aligned} B_{\epsilon_F} &= \left(S_{F,1}\epsilon_F, S_{F,2}\epsilon_F, \ldots, S_{F,d}\epsilon_F\right)_{d\times d}, \\ S_{F,i} &= \text{diag}\left(\frac{\partial \sigma_F^1}{\partial z_i}, \frac{\partial \sigma_F^2}{\partial z_i}, \ldots, \frac{\partial \sigma_F^d}{\partial z_i}\right)_{d\times d}, \quad \text{and} \\ B_{\epsilon_D} &= \left(S_{D,1}\epsilon_D, S_{D,2}\epsilon_D, \ldots, S_{D,d}\epsilon_D\right)_{K\times d}, \\ S_{D,i} &= \text{diag}\left(\frac{\partial \sigma_D^1}{\partial z_i'}, \frac{\partial \sigma_D^2}{\partial z_i'}, \ldots, \frac{\partial \sigma_D^d}{\partial z_i'}\right)_{K\times K}. \end{aligned}$$

This yields

$$\begin{aligned} \mathbb{E}\left[J_{F\circ D}^T J_{F\circ D}\right] &= \mathbb{E}\left[(J_{\mu_D} J_{\mu_F} + J_{\mu_D} B_{\epsilon_F} + B_{\epsilon_D} J_{\mu_F} + B_{\epsilon_D} B_{\epsilon_F})^T (J_{\mu_D} J_{\mu_F} + J_{\mu_D} B_{\epsilon_F} + B_{\epsilon_D} J_{\mu_F} + B_{\epsilon_D} B_{\epsilon_F})\right] \\ &= J_{\mu_F}^T J_{\mu_D}^T J_{\mu_D} J_{\mu_F} + \mathbb{E}\left[B_{\epsilon_F}^T J_{\mu_D}^T J_{\mu_D} B_{\epsilon_F}\right] + \mathbb{E}\left[J_{\mu_F}^T B_{\epsilon_D}^T B_{\epsilon_D} J_{\mu_F}\right] + \mathbb{E}\left[B_{\epsilon_F}^T B_{\epsilon_D}^T B_{\epsilon_D} B_{\epsilon_F}\right] \\ &= J_{\mu_F}^T \left(J_{\mu_D}^T J_{\mu_D} + \mathbb{E}\left[B_{\epsilon_D}^T B_{\epsilon_D}\right]\right) J_{\mu_F} + \mathbb{E}\left[B_{\epsilon_F}^T \left(J_{\mu_D}^T J_{\mu_D} + B_{\epsilon_D}^T B_{\epsilon_D}\right) B_{\epsilon_F}\right] \end{aligned}$$

where in the second equality we used the fact that ϵ_D, ϵ_F are independent and $\mathbb{E}\left[B_\epsilon\right] = 0$. By Lemma 1 we have

$$\mathbb{E}\left[B_{\epsilon_D}^T B_{\epsilon_D}\right] = J_{\sigma_D}^T J_{\sigma_D}.$$

Similarly,

$$\mathbb{E}\left[B_{\epsilon_F}^T B_{\epsilon_D}^T B_{\epsilon_D} B_{\epsilon_F}\right] = \mathbb{E}\left[\mathbb{E}\left[B_{\epsilon_F}^T B_{\epsilon_D}^T B_{\epsilon_D} B_{\epsilon_F} | \epsilon_F\right]\right] = \mathbb{E}\left[B_{\epsilon_F}^T J_{\sigma_D}^T J_{\sigma_D} B_{\epsilon_F}\right] = J_{\sigma_F}^T \text{diag}\left(J_{\sigma_F}^T J_{\sigma_F}\right) J_{\sigma_F}$$

Finally,

$$\mathbb{E}\left[B_{\epsilon_F}^T J_{\mu_D}^T J_{\mu_D} B_{\epsilon_F}\right] = J_{\sigma_F}^T \text{diag}\left(J_{\mu_D}^T J_{\mu_D}\right) J_{\sigma_F}.$$

This completes the proof. □