Continuous-Time Model-Based Reinforcement Learning

Çağatay Yıldız [1] **Markus Heinonen** [1] **Harri Lähdesmäki** [1]

Abstract

Model-based reinforcement learning (MBRL) approaches rely on discrete-time state transition models whereas physical systems and the vast majority of control tasks operate in continuous-time. To avoid time-discretization approximation of the underlying process, we propose a continuous-time MBRL framework based on a novel actor-critic method. Our approach also infers the unknown state evolution differentials with Bayesian neural ordinary differential equations (ODE) to account for epistemic uncertainty. We implement and test our method on a new ODE-RL suite that explicitly solves continuous-time control systems. Our experiments illustrate that the model is robust against irregular and noisy data, is sample-efficient, and can solve control problems which pose challenges to discrete-time MBRL methods.

1. Introduction

The majority of model-based reinforcement learning (MBRL) methods are based on auto-regressive, discrete-time transition functions that take the current state and action as input and output a new state (or a distribution over it). Since the error incurred by the transition function can be detrimental to learning controls, RL methods have been designed to take *epistemic* uncertainty into account. Two standard ways of doing so have been utilizing ensembles (Kurutach et al., 2018; Chua et al., 2018; Janner et al., 2019; Pan et al., 2020) and parametric uncertainty (Depeweg et al., 2017; Gal et al., 2016).

The most notable deep MBRL breakthroughs have been showcased on discrete-time (DT) problems such as games (Hafner et al., 2021), whereas most of the physical and biological systems in science and engineering are inherently continuous in time and follow differential equation dynamics. Despite this fundamental misalignment, discrete-time MBRL has been often applied to continuous-time problems (such as CartPole) by assuming discretized timesteps (Brockman et al., 2016; Tassa et al., 2018). Indeed, approximating ordinary differential equations (ODE) with the simplest Euler solver where the time increment equals the time difference between measurements matches DT-MBRL and resulting discretized system converges to the true ODE as the discretized time increment approaches zero. Nevertheless, accurate approximations come at the cost of increased computational load, a trade-off that is often overlooked. Also, there is no consensus among practitioners on how to choose the correct discretization stepsizes, or how to account for irregular observation times.

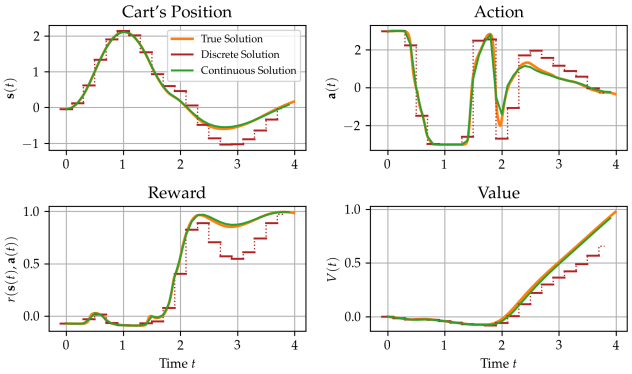

Figure 1: A comparison of true solution of the CartPole system against discrete and continuous-time trajectories. While continuous-time solution almost perfectly matches the true solution, discrete trajectory diverges from it.

Aside from discretization issue, adopting continuous-time reinforcement learning setting (Bradtke and Duff, 1994; Doya, 2000; Frémaux et al., 2013) presents both conceptual and algorithmic challenges. First, Q-functions are known to vanish in continuous-time systems (Baird, 1994; Tallec et al., 2019), which prevents even simple conventional actor-critic or policy learning methods (Sutton and Barto, 2018) in such a setting. Second, the auto-regressive one-step ahead state transitions need to be replaced with time derivatives. This has been demonstrated in physics-informed domains such as Lagrangian (Lutter et al., 2019) or Hamiltonian mechanics (Zhong et al., 2020), or in non-RL domains such the pioneering work of neural ODE (NODE) (Chen et al., 2018). Inference of general-purpose dynamics using ODEs in control settings is still an open question.

[1]Department of Computer Science, Aalto University, Finland. Correspondence to: Çağatay Yıldız <cagatay.yildiz@aalto.fi>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

In this work, we present a continuous-time model-based reinforcement learning paradigm that carefully addresses all challenges above. Our approach can learn arbitrary time differentials of the governing real-world dynamics, and then forward simulates the surrogate ODE dynamics to learn optimal policy functions. Our key contributions are:

- We present a truly continuous-time RL engine, which inherits numerical guarantees of ODE solvers and allows investigation of real-world problems with computer simulations [1].
- We describe novel strategies to handle epistemic uncertainty in neural ODE models and successfully infer black-box controlled ODEs.
- We present a novel theoretically consistent CT actor-critic algorithm that generalizes existing state-value based policy learning methods.
- We demonstrate our method on three different RL problems and present its robustness to both noisy and irregular data, which poses major challenges to conventional discrete-time methods.

2. Continuous-Time Reinforcement Learning

We consider continuous-time (CT) control systems governed by an ordinary differential equation (ODE)

$$\dot{\mathbf{s}}(t) = \frac{d\mathbf{s}(t)}{dt} = \mathbf{f}\big(\mathbf{s}(t), \mathbf{a}(t)\big), \tag{1}$$

where $\mathbf{s}(t) \in \mathbb{R}^d$ denotes a vector-valued *state* function at time t, and $\mathbf{a}(t) \in \mathbb{R}^m$ is the *control* input function, or action. The function $\mathbf{f} : \mathbb{R}^{d+m} \mapsto \mathbb{R}^d$ maps the current state-action pair $(\mathbf{s}, \mathbf{a})$ to a state derivative $\dot{\mathbf{s}}$.

The state at real-valued time $t \in \mathbb{R}_+$ depends on the initial state $\mathbf{s}_0$ as well as the infinitesimal sequence of past actions $\mathbf{a}[0, t)$, and can be solved with

$$\mathbf{s}(t) = \mathbf{s}_0 + \int_0^t \mathbf{f}\big(\mathbf{s}(\tau), \mathbf{a}(\tau)\big) d\tau,$$

where $\tau \in \mathbb{R}_+$ is an auxiliary time variable.

The main objective of CT control is to maximize the reward integral $\int_0^\infty r\big(\mathbf{s}(\tau), \mathbf{a}(\tau)\big) d\tau$, which we translate instead into maximizing the discounted infinite horizon reward integral (Sutton and Barto, 2018), or *value* function,

$$V(\mathbf{s}(t)) = \int_t^\infty e^{-\frac{\tau - t}{\eta}} r\big(\mathbf{s}(\tau), \mathbf{a}(\tau)\big) d\tau, \tag{2}$$

where $r : \mathbb{R}^{d+m} \to \mathbb{R}$ is an instantaneous reward and $0 < \eta < 1$ is the discount factor. The optimal control sequence $\mathbf{a}^*[t, \infty]$ maximizing $V(\mathbf{s}(t))$ is usually bounded by $[\mathbf{a}_{\min}, \mathbf{a}_{\max}]$.

The control problem (2) is a constrained function optimisation problem, which is generally intractable due to system non-linearity and continuous dynamics (Bertsekas et al., 1995). For very limited cases such as linear state feedback $\mathbf{a} = K\mathbf{s} + \mathbf{v}$, the solution can be analytically obtained by Pontryagin's maximum principle (Athans and Falb, 2013) or dynamic programming (Bellman, 1966; Bertsekas et al., 1995). For an overview of numerical approaches to solve continuous-time optimal control problems, see e.g. (Diehl and Gros, 2017).

In this paper, we approach the control problem from a reinforcement learning standpoint. In particular, our main goal is to find a *policy* function

$$\mathbf{a}(t) = \boldsymbol{\pi}(\mathbf{s}(t)) \tag{3}$$

that maximizes the values $V(\mathbf{s}(t))$. This problem formulation signifies a fundamental separation from standard RL methods in several ways.

- First, for the dynamics differential $\dot{\mathbf{s}}(t)$ (1) to exist, the function $\mathbf{f}$ needs to be differentiable over time. Since the existing MBRL methods typically approximate the unknown true CT dynamics via DT stochastic function approximators such as Gaussian processes (Deisenroth and Rasmussen, 2011), Bayesian neural networks (Gal et al., 2016; Depeweg et al., 2017) or probabilistic neural networks (Chua et al., 2018), they cannot be trivially converted into continuous-time.
- Second, deterministic dynamics is also difficult to match with non-deterministic policies (3). Therefore, standard policy gradient methods that require stochastic policies, such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017), are no longer applicable.
- Finally, since the Q-function is ill-defined in continuous time (Baird, 1994), Q-learning based policy methods, such as DDPG (Lillicrap et al., 2016), SAC (Haarnoja et al., 2018) and MBPO (Janner et al., 2019), are eliminated.

2.1. Earlier works

The earliest work in CTRL literature, named advantage updating, was pioneered by Baird (1993) and later extended by Tallec et al. (2019). This model-free algorithm learns the advantage and state-value functions by discretizing the value integral $V(\cdot)$ (2). The optimal policy is then given by the action that maximizes the advantage function for any input state. In control literature continuous-time systems have been studied under limited linear-quadratic (LQ) control setting (Wang et al., 2020; Modares and Lewis, 2014), while

[1] Our model implementation can be found at https://github.com/cagatayyildiz/oderl

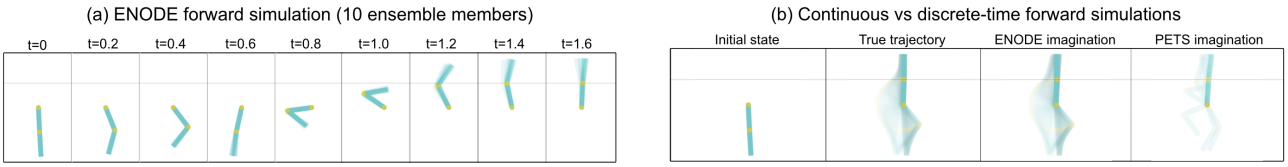

Figure 2: An illustration of our ENODE dynamics model on Acrobot task. **(a)** Ten trajectory samples computed by (7) at uniform intervals. **(b)** The continuous path from initial state to goal state using the true and ENODE dynamics as well as PETS' discrete transition model (single ensemble member is plotted).

Ghavamzadeh and Mahadevan (2001) proposed a CT hierarchical RL method for distributed value functions. Recent work also explores optimal control of physics-informed CT neural dynamical systems such as Lagrangian (Lutter et al., 2019) and Hamiltonian mechanics (Zhong et al., 2020). Finally, Du et al. (2020) present a CT-MBRL framework for semi-Markov decision processes (MDPs) with a latent ODE-RNN dynamics model and a Q-learning approach adapted for semi-MDPs. A through overview of related work is given in supplementary Section S4.

The closest to our work is the seminal paper by Doya (2000), which describes a technique to learn optimal policy in continuous-time while respecting above-mentioned limitations. The method approximates the state-value function with a radial basis function and Euler discretization of the value integral (2). An actor-critic algorithm based on temporal difference error was also proposed. Frémaux et al. (2013) adapt the model to spiking neural networks.

Our approach, outlined in Algorithm 1, differs fundamentally from these approaches in two respects: *(i)* we estimate the unknown dynamics using non-linear neural ODEs, and *(ii)* explicitly forward integrate the dynamics for non-linear policy learning. Next we detail the components of our model, namely, the dynamics model and continuous-time actor-critic algorithm.

3. Dynamics Learning

We start by assuming that the dataset consists of N observed state-action trajectories $\mathcal{D} = \{(\mathbf{s}_{0:T}^{(n)}, \mathbf{a}_{0:T}^{(n)})\}_{n=1}^{N}$ collected from a CT environment we aim to model or, in case of simulated environments, obtained by solving the true ODE system at observation time points $t_{0:T}$. We denote the i'th observation within a trajectory by $\mathbf{s}_i := \mathbf{s}(t_i)$, and $\Delta t_i = t_{i+1} - t_i$ stands for the time until the next observation $\mathbf{s}_{i+1}$. We note that observations may arrive irregularly in time.

3.1. Previous Work

The dynamics models in state-of-the-art MBRL techniques such as PILCO (Deisenroth and Rasmussen, 2011) and PETS (Chua et al., 2018) approximate the state difference $\Delta\mathbf{s}_t = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)$ such that $\mathbf{s}_{t+1} = \mathbf{s}_t + \Delta\mathbf{s}_t$. Such approximations are implicitly based on uniform sampling assumption. To see that, imagine the system visits some state twice $\mathbf{s}_i = \mathbf{s}_j = \mathbf{s}$ at two timepoints t_i and t_j, and the same action $\mathbf{a}$ is applied. If the observations arrive non-uniformly, i.e., $\Delta t_i \neq \Delta t_j$, then the next states would be different: $\mathbf{s}_{i+1} \neq \mathbf{s}_{j+1}$ (given the trivial condition $||\mathbf{f}(\mathbf{s})|| \neq 0$). Consequently, resulting data triplets for dynamics learning would have the same inputs $(\mathbf{s}, \mathbf{a})$ and different targets, which would lead to training issues.

To adapt these discrete-time methods for temporally irregular trajectories, we give the time increment Δt_i as an additional parameter to the dynamics transition,

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{f}(\mathbf{s}_i, \mathbf{a}_i, \Delta t_i). \tag{4}$$

We refer this version with *modified* prefix and use *vanilla* for the traditional way of training.

3.2. CT Dynamics with Bayesian Neural ODEs

We consider deep neural ODE models $\hat{\mathbf{f}}_{\boldsymbol{\theta}}$ (Chen et al., 2018), where $\boldsymbol{\theta}$ denotes the network parameters. Our goal is to estimate the state differential $\hat{\mathbf{f}}_{\boldsymbol{\theta}}$ such that its forward-simulated ODE trajectories reproduce the true ODE trajectories accurately. Our main learning objective is to compute the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. Unfortunately, the exact posterior in black-box ODE models are intractable and typically MAP estimates are computed (Heinonen et al., 2018; Chen et al., 2018). Since handling epistemic uncertainty to reduce model bias lies at the heart of MBRL (Deisenroth and Rasmussen, 2011; Chua et al., 2018), we resort to variational inference by introducing an approximate posterior distribution $q(\boldsymbol{\theta})$ closest to the true posterior in KL sense,

$$\arg\min_{q(\boldsymbol{\theta})} \mathrm{KL}[q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta}|\mathcal{D})].$$

Finding the optimal variational approximation is equivalent to maximizing the evidence lower bound (ELBO) (Blei et al., 2017):

$$\log p(\mathcal{D}) \geq \mathbb{E}_q\Big[\log p(\mathcal{D}|\boldsymbol{\theta})\Big] - \mathrm{KL}\left[q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta})\right], \tag{5}$$

where $p(\boldsymbol{\theta})$ is the parameter prior. Note that the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ is evaluated only on the state-trajectories,

Algorithm 1 Continuous-Time MBRL with Neural ODEs

Choose a variational approximation for NODE model and initialize dynamics $\mathbf{f}_{\boldsymbol{\theta}}$, actor $\boldsymbol{\pi}_{\boldsymbol{\phi}}$ and critic $\nu_{\boldsymbol{\xi}}$ estimates
Collect a dataset $\mathcal{D} = \{(\mathbf{s}_{0:T}^{(n)}, \mathbf{a}_{0:T}^{(n)})\}_{n=1}^{N}$ of N trajectories with smooth random policies $\pi^{(n)}(\mathbf{s}, t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{k}(t, t'))$
repeat
  `// Dynamics learning`
  for $i = 1 ... N_{\text{dyn}}$ **do**
    Randomly sample from $\mathcal{D}$ a mini-batch of N_{d} subsequences, each of which of length $t_{\text{s}} = 5$: $\mathcal{D}' = \{(\mathbf{s}_{t_n:t_n+t_s}^{(n)}, \mathbf{a}_{t_n:t_n+t_s}^{(n)})\}_{n \in \Xi}, \;\; \Xi \subset \{1, \ldots, N\}, \;\; |\Xi| = N_{\text{d}}, \;\; t_n \sim \mathcal{U}[0, T - t_s]$
    Update $\boldsymbol{\theta}$ by taking a gradient step using the ELBO on $\mathcal{D}'$ by (5,6,7)
  end for
  `// Actor-Critic learning`
  for $i = 1 ... N_{\text{ac}}$ **do**
    Uniformly sample $N_p = 100$ states $\{\mathbf{s}^{(p)}\}_{p=1}^{N_p}$ from the most recently collected ten data sequences
    Imagine trajectories $\hat{\mathbf{s}}$ from each $\mathbf{s}^{(p)}$ using current estimates of dynamics $\hat{\mathbf{f}}_{\boldsymbol{\theta}}$ and policy $\boldsymbol{\pi}$ by (9,10).
    Update $\boldsymbol{\phi}$ and $\boldsymbol{\xi}$ by taking gradient steps using the imagined trajectories by (13,14)
  end for
  Collect data $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t)_{t=0}^{T}\}$ by interacting with real world: $\mathbf{s}_{0:T}, \mathbf{a}_{0:T} \leftarrow \mathbf{s}_0 + \int_0^T \mathbf{f}_{\text{true}}(\mathbf{s}(\tau), \boldsymbol{\pi}(\mathbf{s}(\tau)))d\tau$
  [optional] Collect more sequences with an exploring policy: $\boldsymbol{\pi}^{\text{exp}}(\mathbf{s}(t), t) = \boldsymbol{\pi}(\mathbf{s}(t)) + \mathbf{z}(t), \;\; \mathbf{z}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{k}(t, t'))$
  Execute the policy in the real environment with ten different initial configurations
until The pole never falls once upright in ten trials

i.e., $p(\mathcal{D}|\boldsymbol{\theta}) = p(\mathcal{D}_{\mathbf{s}}|\mathcal{D}_{\mathbf{a}}, \boldsymbol{\theta})$, where $\mathcal{D}_{\mathbf{s}} = \{\mathbf{s}_{0:T}^{(n)}\}_{n=1}^{N}$ and $\mathcal{D}_{\mathbf{a}} = \{\mathbf{a}_{0:T}^{(n)}\}_{n=1}^{N}$. An unbiased estimate of the expected log-likelihood can be obtained by Monte Carlo integration,

$$\mathbb{E}_q\big[\log p(\mathcal{D}|\boldsymbol{\theta})\big] \approx \frac{1}{L}\sum_{l=1}^{L}\sum_{n=1}^{N}\sum_{i=0}^{T}\log\mathcal{N}\left(\mathbf{s}_i^{(n)}|\hat{\mathbf{s}}_i^{(n,l)}, \Sigma\right), \tag{6}$$

where Σ denotes trainable observation noise variances. A trajectory sample $\hat{\mathbf{s}}_{0:T}^{(n,l)}$ is drawn by first sampling from the posterior $\boldsymbol{\theta}^{(l)} \sim q(\boldsymbol{\theta})$ and then forward simulating the dynamics model $\hat{\mathbf{f}}_{\boldsymbol{\theta}^{(l)}}(\mathbf{s}, \mathbf{a})$:

$$\hat{\mathbf{s}}_i^{(n,l)} \mid \left(\mathbf{s}_0^{(n)}, \mathbf{a}_{0:t}^{(n)}\right) = \mathbf{s}_0^{(n)} + \int_0^{t_i}\hat{\mathbf{f}}_{\boldsymbol{\theta}^{(l)}}\left(\hat{\mathbf{s}}_\tau^{(n,l)}, \mathbf{a}_\tau^{(n)}\right)d\tau. \tag{7}$$

We demonstrate trajectory samples in Figure 2a.

Time-continuous actions The state trajectory in (7) is fully determined by the initial state $\mathbf{s}_0$ and the infinitesimal actions $\mathbf{a}[0, t)$ applied during integration. Since the integral may be evaluated at arbitrary time points, the actions must be continuous in time, whereas they are assumed to be known only at observation time points. A simple yet convenient mean to obtain time-continuous actions is interpolating between observed discrete actions. We opt for kernel interpolation with a squared exponential kernel function whose length-scale parameter can be set depending on the smoothness of the action sequence. GP-based stochastic interpolations would also be considered to reduce the bias incurred by interpolation.

Variational approximation If the variational posterior is chosen to be a mixture of Dirac densities on the weights, the resulting approximation corresponds to an ensemble of neural ODEs (ENODE). Similar to other ensemble methods, each member would learn a different vector field, which provide means for epistemic uncertainty modeling. Alternatively, $q(\boldsymbol{\theta})$ can be defined on the weights, hidden neurons or function outputs, which would lead to weight-space (Yildiz et al., 2019), implicit (Trinh et al., 2020) or functional (Sun et al., 2019) Bayesian neural ODE models.

4. Continuous-Time Actor-Critic Algorithm

In this section, we describe an actor-critic algorithm that is fully compatible with the CT framework. Our algorithm relies solely on the forward simulation of the inferred dynamics, i.e., no interaction with the real-world is required, making it much easier to deploy to real-world compared to the standard MB controllers such as MPC (Chua et al., 2018) and policy learning methods such as MBPO (Janner et al., 2019). Also, the standard discrete-time temporal difference (TD) learning methods (Schulman et al., 2015; Sutton and Barto, 2018; Hafner et al., 2020) appear as special cases of our CT actor-critic method as shown later.

Our main policy learning goal is to maximize the discounted cumulative reward when the policy is executed in the real-world, which simply amounts to computing the integral in (2). Because the true differential function is unknown, we leverage our proxy dynamics $\hat{\mathbf{f}}_{\boldsymbol{\theta}}$ to maximize a surrogate objective:

$$V(\mathbf{s}_0) \approx \hat{V}(\mathbf{s}_0) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\int_0^\infty e^{-\frac{\tau}{\eta}} r(\hat{\mathbf{s}}(\tau), \hat{\mathbf{a}}(\tau)) d\tau\right] \quad (8)$$

where $\hat{\mathbf{s}}[0,t)$ and $\hat{\mathbf{a}}[0,t)$ are referred to as *imagined trajectory* and *imagined actions*[2] and $\mathbf{s}_0$ denotes an initial value. The imagination is controlled by a deterministic policy function $\boldsymbol{\pi}_{\boldsymbol{\phi}}(\cdot)$ with parameters $\boldsymbol{\phi}$:

$$\hat{\mathbf{s}}(t) = \mathbf{s}_0 + \int_0^t \hat{\mathbf{f}}_{\boldsymbol{\theta}}\big(\hat{\mathbf{s}}(\tau), \hat{\mathbf{a}}(\tau)\big) d\tau \quad (9)$$

$$\hat{\mathbf{a}}(t) = \boldsymbol{\pi}_{\boldsymbol{\phi}}(\hat{\mathbf{s}}(t)). \quad (10)$$

Note that this formulation comes with the differentiable reward assumption. The reward function typically consists of a sum $r(\mathbf{s}, \mathbf{a}) = d(\mathbf{s}, \mathbf{s}^*) + c||\mathbf{a}||_2$, where the first term measures some distance between the current state and a target state $\mathbf{s}^*$, and the latter term penalizes the action magnitude. In turn, policy learning task can be informally stated as finding a set of policy parameters $\boldsymbol{\phi}$ that drives the system towards high-reward states while applying smallest possible actions. If the initial state $\mathbf{s}_0$ follows a distribution $\mathbf{s}_0 \sim p(\mathbf{s}_0)$, then the optimization target becomes $\mathbb{E}_{p(\mathbf{s}_0)}[\hat{V}(\mathbf{s}_0)]$, which is usually approximated by Monte-Carlo averaging.

Critic The optimization objective in (8) requires computing an infinite integral. To circumvent this problem, we first re-write the value function in a recursive manner:

$$\hat{V}^H(\mathbf{s}_0) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\int_0^H e^{-\frac{\tau}{\eta}} r(\hat{\mathbf{s}}(\tau), \hat{\mathbf{a}}(\tau)) d\tau + e^{-\frac{H}{\eta}} \hat{V}(\hat{\mathbf{s}}_H)\right]$$

where H is known as *horizon*, $\hat{\mathbf{s}}_H := \hat{\mathbf{s}}(H)$ is the end-state of the integral, and $\hat{V}(\hat{\mathbf{s}}_H)$ is *future reward*. The first term on the rhs represents the imagined reward, which can be computed by evaluating the reward function on the trajectories given by (9-10), whereas the second term is an infinite integral. Next we introduce the critic, or state-value approximation, that replaces intractable future reward term:

$$\hat{V}(\mathbf{s}) \approx \nu_{\boldsymbol{\xi}}(\mathbf{s}), \quad (11)$$

where $\nu_{\boldsymbol{\xi}}$ is a neural network with parameters $\boldsymbol{\xi}$. With a slight abuse of notation, we rewrite the value function as follows:

$$\hat{V}^H(\mathbf{s}_0) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\int_0^H e^{-\frac{\tau}{\eta}} r(\hat{\mathbf{s}}(\tau), \hat{\mathbf{a}}(\tau)) d\tau + e^{-\frac{H}{\eta}} \nu_{\boldsymbol{\xi}}(\hat{\mathbf{s}}_H)\right]. \quad (12)$$

Consequently, our new objective seeks to learn a policy function $\boldsymbol{\pi}_{\boldsymbol{\phi}}$ that maximizes the reward integral over horizon H and ends up at a high-value state.

[2]The actions are applied to imagined states, rendering the actions imagined as well.

Learning the actor The actor aims to maximize the value estimates (12) for a batch of initial values. In order to learn the optimal policy everywhere, initial values are drawn from the previous experience. More formally, we solve the following maximization problem:

$$\max_{\boldsymbol{\phi}} \mathbb{E}_{q(\mathbf{s})}\left[\hat{V}^H(\mathbf{s})\right], \quad q(\mathbf{s}) = \frac{1}{|\mathcal{D}|}\sum_{n,i} \delta\left(\mathbf{s} - \mathbf{s}_i^{(n)}\right). \quad (13)$$

Observe that both the reward integral and the future reward in (12) depend on the imagined state trajectory, which in turn depends on the actions. Therefore, the gradient of (13) w.r.t. $\boldsymbol{\phi}$ can easily be computed in auto-differentiation frameworks that support ODE gradients.

Learning the critic Similar to discrete-time frameworks, our method utilizes temporal difference learning to estimate the state-value function in (11). In particular, the value approximation $\nu_{\boldsymbol{\xi}}(\cdot)$ is regressed on the sum of the finite reward integral and future reward given in (12). Since the value approximation holds for any horizon length H, we propose to marginalize it out:

$$\min_{\boldsymbol{\xi}} \mathbb{E}_{q(\mathbf{s})}\left[\left(\nu_{\boldsymbol{\xi}}(\mathbf{s}) - \mathbb{E}_{q(h)}\left[\hat{V}^h(\mathbf{s})\right]\right)^2\right], \quad q(h) = \mathcal{U}[0, H] \quad (14)$$

where the outer expectation is w.r.t. $q(\mathbf{s})$ shown in (13), $\mathcal{U}$ denotes the uniform distribution, and the inner expectation is approximated by Monte-Carlo sampling. We use a separate target network that generates the future rewards, an idea that is known to stabilize the critic training (Mnih et al., 2015). We update the target network with a copy of the critic every 100 optimization iterations.

Both actor and critic loss functions require trajectory sampling from the dynamics model. In practice, we forward integrate the dynamics model once and utilize the resulting state-action trajectories to optimize both objectives. The inner expectation in (14) can be evaluated using the intermediate states without incurring any additional overhead.

Connection to temporal difference Discrete state-value formulation can be retrieved by discretizing the reward integral (2) with a predetermined step size $\overline{V}(\mathbf{s}_t) = \sum_{l=0}^{\infty} \gamma^l r(\mathbf{s}_{t+l}, \mathbf{a}_{t+l})$ where the overbar sign denotes discrete formulation and γ is the discount factor. In such a scenario, $\hat{V}^h(\mathbf{s})$ would reduce to so-called h-step return: $\overline{V}^h(\mathbf{s}_t) = \sum_{l=0}^{h} \gamma^l r_{t+l} + \overline{V}(\mathbf{s}_{t+h})$. Marginalizing the horizon would correspond to computing the mean of the h-step returns, a commonly used technique to reduce the gradient variance (Schulman et al., 2016). Finally, TD-h learning would be retrieved by simply keeping the horizon fixed at h, i.e., with $q(h)$ being a Dirac distribution. As demonstrated

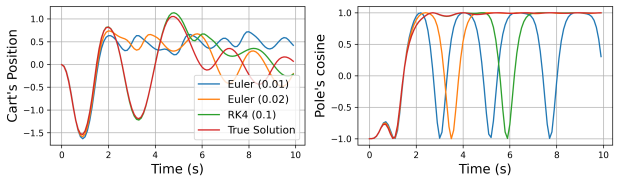

Figure 3: Comparison of different discretization choices from (Brockman et al., 2016; Gal et al., 2016; Tassa et al., 2018) on CartPole swing-up task vs. the true ODE solution. Discretization timesteps are given inside the parenthesis.

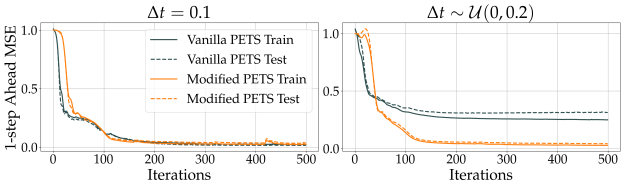

Figure 4: Comparison of the vanilla and modified versions of PETS dynamics on regularly and irregularly sampled data. Vanilla PETS cannot handle irregularly sampled data while model performances match on the former case.

in Figure 2b, DT models compute the value estimates using a finite collection of states $\mathbf{s}_{t:t+h}$, and thus are inaccurate.

5. Experiments

We experimentally evaluate our model on three CTRL benchmark environments: Pendulum, CartPole and Acrobot. In all tasks, the initial state is "hanging down" and the goal is to swing up and stabilise the pole at the target position where the pole stays in an upright position (also at the origin in CartPole task). We define the differentiable reward functions as the exponential of the distance to target state, while also penalizing the magnitude of the velocity and action. The actions are assumed to be continuous and bounded. The experiments aim to answer the following questions:

- Do existing RL frameworks simulate CT dynamics?
- Do DTRL methods fail with irregularly sampled data?
- Is our ENODE approximation sensitive to the imagination horizon and the number of ensemble members?
- How does our model perform in standard RL tasks?
- How does the model performance change with *(i)* noisy data, *(ii)* irregular sampling, and *(iii)* various Δt?

Compared models We consider three different variational approximations for our model: Ensemble neural ODE (ENODE), batch ensemble neural ODE (BENODE) and implicit Bayesian neural ODE (IBNODE). ENODE maintains a separate MLP for each ensemble member (Lakshminarayanan et al., 2017) while BENODE learns shared weight parameters as well as member-specific rank-1 multipliers (Wen et al., 2020). IBNODE considers a mixture of Gaussians variational posterior on layer-wise multiplicative factors (Trinh et al., 2020). In addition, we include two DT transition models, namely, modified PETS and deep PILCO (Gal et al., 2016). Policy learning using DT transition models is performed by discretizing the integrals in (13,14).

Observation spacing In standard RL frameworks such as OpenAI Gym, the observations arrive at constant intervals $\Delta t = \kappa$ (Brockman et al., 2016). In addition to this baseline, we consider two irregular sampling scenarios in which observations *(i)* arrive uniformly within a range $\Delta t \sim \mathcal{U}(0, 2\kappa]$, or *(ii)* follow an exponential distribution $\text{Exp}(\kappa)$ with κ being the scale parameter. Note that all distributions have the same mean time increments κ.

Additional details We refer to each iteration of the while loop in Algorithm 1 as a *round*. Following Deisenroth and Rasmussen (2011), the model performance is measured at the end of each round by executing the inferred policy in the real world for $H = 30$ seconds. Our main result Figures 5-6-7 illustrate mean rewards and 90/10 percent quantiles, computed over 20 experiments. Each method is given the same amount of computational budget; therefore, the number of completed rounds differ among methods. We implement our method in PyTorch (Paszke et al., 2019) and use adaptive checkpoint adjoint method to backpropagate through the ODE solver (Zhuang et al., 2020). Please refer to supplementary Section S2 for more details.

5.1. A New CTRL Framework

The test environments are already implemented in standard RL frameworks such as OpenAI Gym (Brockman et al., 2016) and DeepMind Control Suite (Tassa et al., 2018). Existing implementations either perform discrete transitions or inaccurate numerical integration routines, e.g., fixed-step ODE solvers with too big step sizes. Furthermore, as illustrated in Figure 1, the actions in our CT framework smoothly change over time whereas DT approximations implicitly assume piece-wise constant control signals. Figure 3 illustrates how different discretization choices result in distinct state trajectories on CartPole problem: While the policy is able to swing-up and balance the pole in the upright position when integrated with the correct solver, crude Euler and RK4 approximations bifurcate near the position where the pole hangs up.

To deal with above-mentioned inaccuracies, we implement a new CTRL framework which takes as input a deterministic policy function $\boldsymbol{\pi}(\cdot)$ and observation time points, performs forward integration in a numerically-stable way, and returns the state trajectory at corresponding time points. Empirical

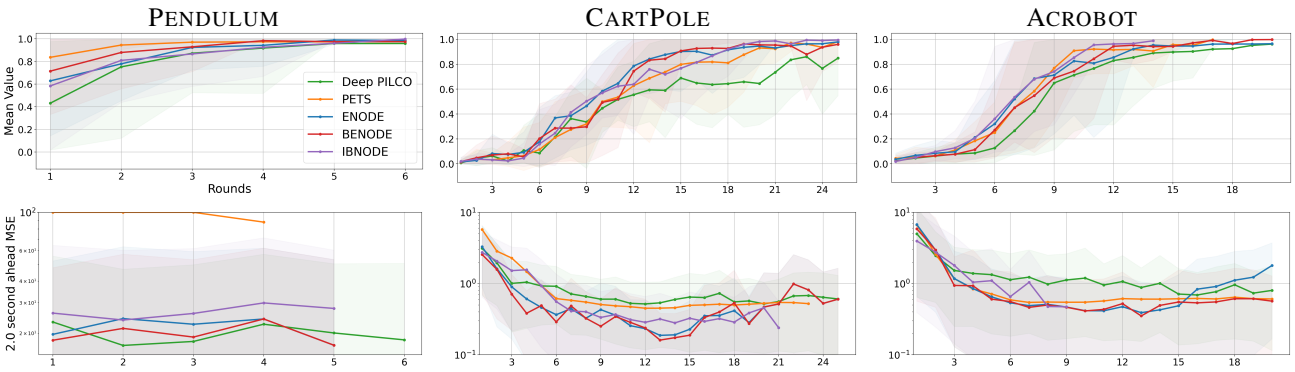

Figure 5: A comparison of three variants of our model and two other dynamics approximations (Gal et al., 2016; Chua et al., 2018) on a dataset of noise-free and regularly spaced observations with $\Delta t = 0.1$. Top row illustrates the mean reward curves (computed for $H = 30$ seconds) and 90/10 quantiles over rounds and the predictive mean squared error of the dynamics on a test set is visualized over rounds on the bottom row. We observe that all models perform similarly despite the discrepancy in the dynamics accuracy.

evidence for the need of integration as well as a comparison of different ODE solvers are presented in supplementary Section S3.

5.2. Vanilla vs. Modified PETS

In this experiment, we investigate whether vanilla PETS can accurately infer the dynamics when the data sequences are temporally irregular. For this, both versions of PETS are trained on two CartPole datasets with observations arriving at fixed ($\Delta t = 0.1$) or uniformly sampled ($\Delta t \sim \mathcal{U}(0, 0.2]$) increments. The train and test one-step ahead prediction errors are illustrated in Figure 4. Both models successfully learn the transitions when they arrive at fixed increments. Modified PETS achieves near-zero error on temporally irregular dataset as well whereas the vanilla version cannot perfectly fit the training data as conjectured in Section 3. We proceed with modified PETS (MPETS) in our irregular sampling experiments.

5.3. ENODE Hyperparameter Sensitivity

As an ablation study, we check how the imagination horizon H and the number of ensemble members n_{ens} impact the model performance. We consider a noise-free CartPole environment with fixed observation time increments ($\Delta t = 0.1$), and disjointly vary $H \in [1.0, 4.0]$ and $n_{\text{ens}} \in \{1, 3, 5, 10, 15\}$. The mean reward values over rounds are plotted in Figure 6. We observe that our ENODE approximation is somewhat robust to n_{ens}. As expected, too short or long horizons deteriorate the performance. In the remainder of this paper, we fix $H = 2.0$ and $n_{\text{ens}} = 10$.

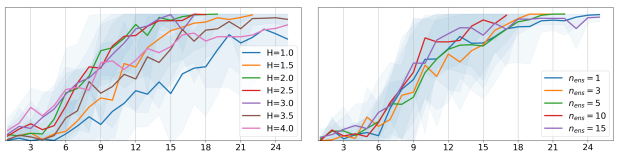

Figure 6: Ablation studies showing our ENODE approximation's sensitivity to the imagination horizon H and the number of ensemble members n_{ens}.

5.4. Comparisons on Standard Setup

We first compare our model variants, Deep PILCO and PETS on Pendulum, CartPole and Acrobot environments with noise-free and regularly sampled observations ($\Delta t = 0.1$). The top tow in Figure 5 illustrates the mean values obtained by executing policies in the real world. We observe that all our model variants as well as PETS perform comparably while Deep PILCO requires more rounds to solve the problems.

Next, we examine how the model performance is related to the dynamics accuracy. For this, we take a snapshot of all models after each round, and also collect a separate test dataset by executing the policies that solve the problems in the real world. The bottom panels in Figure 5 show how the dynamics errors evolve as Algorithm 1 proceeds. Specifically, we plot two-second ahead prediction error, which also equals to the imagination horizon in our Actor-Critic algorithm. Since PETS errors explode in early rounds, we clip it to 100 for visualization purposes. Our method attains high accuracy only after a few rounds whereas the policy converges to optimality much later. We conjecture that dynamics accuracy does not solely predict the overall performance. Additional results demonstrating shorter term predictions can be found in supplementary Figure S2.

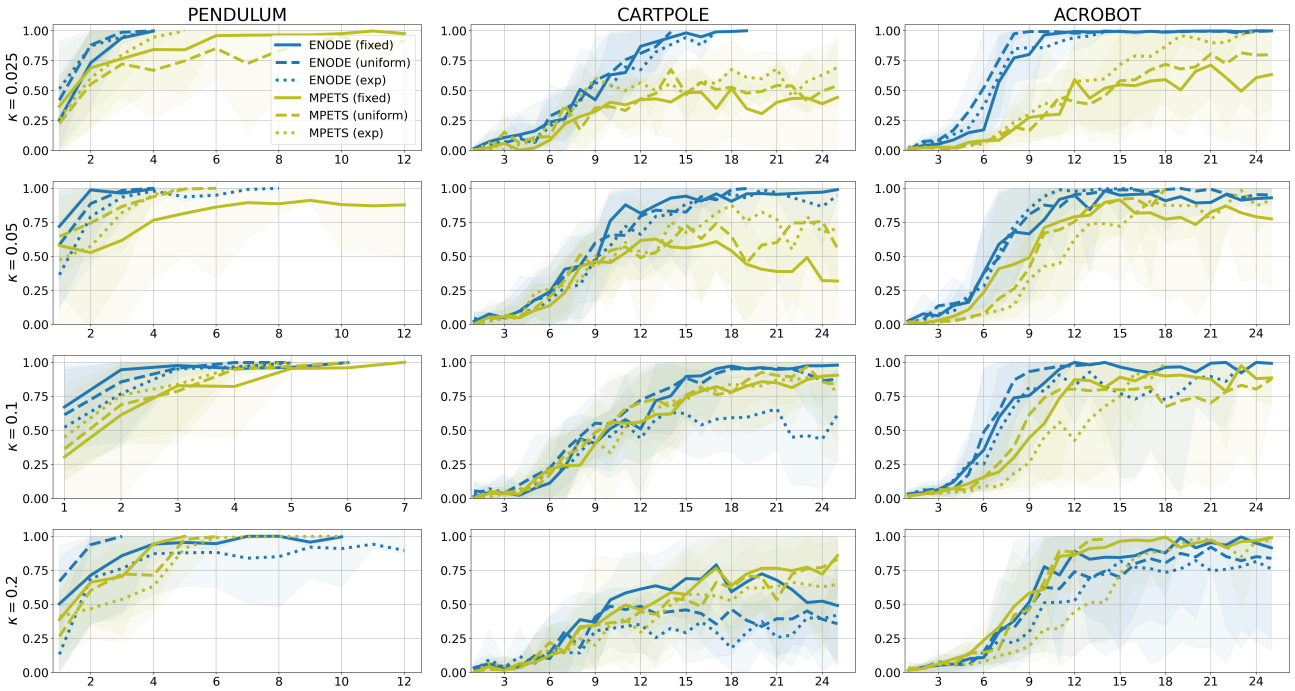

Figure 7: A comparison of ENODE and MPETS on three noisy environments ($\sigma = 0.025$) with three different observation spacings and varying mean time differences κ (note that all three spacings have the same mean time increments κ). We observe that MPETS' performance deteriorates with reduced Δt while our method is more robust to these changes.

5.5. Noisy and Temporally Irregular Data Experiments

To contrast the robustness of discrete and continuous-time techniques, we evaluate MPETS and ENODE on more realistic datasets of irregularly sampled and noisy data sequences, where the observations are perturbed by Gaussian noise with standard deviation $\sigma = 0.025$. We also vary the mean observation time differences $\kappa \in [0.025, 0.05, 0.1, 0.2]$ for all three observation spacings.

Figure 7 exhibits mean value curves on three tasks of our interest. The most striking observation is that small time differences lead to the breakdown of MPETS whereas our model is more tolerant. We believe MPETS' performance drop is inevitable since noise corrupts both the inputs and outputs of the transition function. On the other hand, our method infers the underlying process and thus noise can be handled by the observation noise model without corrupting the dynamics inference. Furthermore, the performances of both models stay roughly unaffected with stochastic time increments. We conclude by highlighting the fact that both models solve the tasks in fewer rounds when the dataset is noise-free and the models perform comparably well.

6. Discussion

We have presented a novel continuous-time MBRL framework that resolves the time discretization approximation of existing techniques. Our dynamics learning method relies on the neural ODE framework, and thus can learn arbitrarily complicated controlled ODE systems. We propose several original strategies to handle epistemic uncertainty in NODE. Furthermore, we develop a new actor-critic algorithm that operates in continuous time and space. We experimentally demonstrate that our method is robust to environment changes such as observation noise, temporal sparsity and irregularity.

References

Michael Athans and Peter Falb. *Optimal control: an introduction to the theory and its applications*. Dover Publications, 2013.

Leemon Baird. Advantage updating. Technical report, Wright Lab, 1993.

Leemon Baird. Reinforcement learning in continuous time: Advantage updating. In *IEEE International Conference on Neural Networks*, volume 4, pages 2448–2453, 1994.

Richard Bellman. Dynamic programming. *Science*, 153 (3731):34–37, 1966.

Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming*

and optimal control, volume 1. Athena scientific Belmont, MA, 1995.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Steven Bradtke and Michael Duff. Reinforcement learning methods for continuous-time Markov decision problems. *NIPS*, 7:393–400, 1994.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv*, 2016.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *NIPS*, pages 6571–6583, 2018.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NIPS*, pages 4754–4765, 2018.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *ICML*, pages 465–472, 2011.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. In *ICLR*, 2017.

Moritz Diehl and Sébastien Gros. *Numerical Optimal Control*. University of Freiburg, 2017.

Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19805–19816. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e562cd9c0768d5464b64cf61da7fc6bb-Paper.pdf.

Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLOS Computational Biology*, 9:1–21, 2013.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *ICML Data-Efficient Machine Learning Workshop*, page 34, 2016.

Mohammad Ghavamzadeh and Sridhar Mahadevan. Continuous-time hierarchical reinforcement learning. In *ICML*, pages 186–193, 2001.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *ICLR*, 2021.

Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ODE models with Gaussian processes. In *ICML*, pages 1959–1968, 2018.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NIPS*, pages 12519–12530, 2019.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *ICLR*, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *ICLR*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Hamidreza Modares and Frank L Lewis. Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning. *IEEE Transactions on Automatic control*, 59(11):3051–3056, 2014.

Feiyang Pan, Jia He, Dandan Tu, and Qing He. Trust the model when it is confident: Masked model-based actor-critic. In *NeurIPS*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, 2017.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian neural networks. In *ICLR*, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep Q-learning methods robust to time discretization. In *ICML*, 2019.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *CoRR*, 2018.

Trung Trinh, Samuel Kaski, and Markus Heinonen. Scalable Bayesian neural networks by layer-wise input augmentation. *arXiv*, 2020.

Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(198):1–34, 2020.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2020.

Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks. In *NIPS*, pages 13412–13421, 2019.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ODE-net: Learning Hamiltonian dynamics with control. In *ICLR*, 2020.

Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ode. In *International Conference on Machine Learning*, pages 11639–11649. PMLR, 2020.