Improving Probabilistic Model Based Reinforcement Learning Control in Chaotic Environments with Conditional Normalizing Flows

Mason Lee [1] **Siddharth Diwan** [1]

Abstract

In this paper, we present a Model-Based Reinforcement Learning (MBRL) algorithm named *Conditional Normalizing Flows Monte Carlo Probabalistic Inference for Learning COntrol* (CNF-MC-PILCO). The algorithm relies on Gaussian Processes (GPs) and Conditional Normalizing FLows (CNF) to model the system dynamics and on a Monte Carlo approach to estimate the policy gradient. This defines a framework in which we ablate the choice of the following components: (i) the layers of the CNF neural network (CNF NN), (ii) the context fed into the CNF NN. The combination of the aforementioned aspects affects dramatically the performance of CNF-MC-PILCO. Numerical comparisons in a simulated cart-pole environment show that CNF-MC-PILCO exhibits comparable control performance w.r.t. state-of-the art GP-based MBRL algorithms. CNF-MC-PILCO code is publicly available at https://github.com/masonlee277/mc-pilco-dpf-nf.

1. Introduction

1.1. RL Control

One naive method of navigating chaotic systems (or data) is to use RL Control. Some common RL controls include Model-Free RL (MFRL) such as state-actor-critic models and Model-Based RL (MBRL). However, MFRL suffers from data inefficiency - hundreds of thousands of trials are required independent of policy iteration, value iteration or policy search methods. A prime example is the data inefficiency of MFRL in solving the mountain car problem (Amadio et al., 2022).

MBRL fixes this issue but introduces model bias through its dynamics model (which is a simulation of the world) - errors in predictions compound over time, and MBRL often underperforms as apposed to MFRL approaches (Ward et al., 2019), (Klink, 2021). In MBRL, an extra term is added to the cost function of a minimum-variance controller, whose uncertainty is used to improve model-parameter estimation. However, if there is a small dataset of observed deterministic transitions, only a small subset of the numerous candidate transition functions between states ('path') would be chosen. By choosing a single line from these paths, MBRL locks in the uncertainty. Instead, we would like to somehow train over the entire uncertainty space (or somehow incorporate it wholly) when generating future states.

Another interesting RL Control is to use Physics Models. In the space of chaotic prediction, Sparse Identification of Nonlinear Dynamical systems (SINDy) is a model which estimates ODEs and PDEs from time series data. With the pre-prending of an Autoencoder, AE-SINDy is capable of predicting the variables and equations governing a chatoic system even for high dimensional data. Then, one can use those outputs to predict the future states of particles following those equations. However, SINDy's greatest limitation is that it requires the user to know the general dynamics of the system before hand, for instance, they must know whether the governing equations are first order or second order.

1.2. PILCO

Probabilistic Inference for COntrol (PILCO) was developed as a response to the aforementioned limitations of previous RL Control approaches (Gal et al.). Unlike traditional MBRL, PILCO is an MBRL algorithm which can optimize itself under uncertainty. It learns from a probabilistic dynamics model and samples states using non-parametric Gausian Processes (GP, this is a class of RL Baysian models). The model uncertainty is temporally uncorrelated noise and is incorporated into planning and control. This is then used to approximate inference for policy evaluation, gradient based policy improvement, and parameter updation.

Unfortunately, PILCO's design has some caveats. First is high computational cost - since it is an MBRL algorithm, there are a massive amount of interactions with environment to train to chaotic systems. Additionally, the use of Squared Exponential (SE) kernels and differentiable cost functions introduces smooth properties on the GPs which leads to poor generalization. This compounds with PILCO's approach of forced unimodality - the model performs moment matching in several places. When predicting uncertainty of the next

state over all places, the distribution of states is smoothed over a gaussian (centered around single target state). Since the initial variance of state distributions is high, and due to dependence on many initial conditions, the optimal solution for sampling future states might be multimodal.

2. Background

2.1. MC-PILCO

Monte-Carlo Probabilistic Inference for COntrol (MC-PILCO) overcomes the strong limitations that PILCO has, keeping the focus on data efficiency. We explain it with more detail as it is our baseline model for results.

The goal of the model is to minimize a Monte-Carlo cumulative cost function J on all particles in each state. The particles are propagated through the stochastic model, and a GP estimates the mean and covariance of the conditional distribution derived via moment matching on the initial state distribution x_0.

Then, M particles are sampled from the initial state distribution $p(x_0)$., and each one of the M particles is propagated using the one-step-ahead GP models.

MC-PILCO, in broad terms, consists of the iteration of three main steps, namely, update the GP models, update the policy parameters, and execute the policy on the system.

The learned model of the one step ahead prediction is indeed a GP. But thanks to the particle filters the model does not assume a state evolution that is gaussian like with moment matching (and that is for example it could handle the two modes solutions in the cart pole) (figure?)

Following is the algorithm employed by MC-PILCO (Amadio et al., 2022):

2.2. MC-PILCO Limitations

The use of a stochastic policy during policy optimization allows increasing the entropy of the particles' distribution. This property increments the probability of visiting low-cost regions and escaping from local minima. However, MC-PILCO still gets caught in local minima and cannot escape. We believe there needs to be more exploration in the dynamics. While particle filters are theoretically able to handle bimodal distributions, final objective is to have deterministic policy.

The authors' decision to use particle filters, theoretically capable of managing bimodal distributions, aimed at achieving a deterministic policy. Furthermore, they integrated MC-dropout into their methodology. Still, the issue of local minima persists, signifying that further enhancements to their approach may be required to overcome these intricate dynamics more effectively.

Algorithm 1 MC-PILCO

Input: number of expxlorations N_e, random exploration policy function π_r, number of particles M, number of optimization steps N_{opt} initial policy π_θ, MC cumulative cost J, kernel k,learning rate α_{lr}, minimum learning rate $\alpha_{\text{lr,min}}$, rollout length T, particle dropout p_d, dropout probability reduction Δp_d, and monitoring signal parameters: σ_s, λ_s, n_s.
Notation Remark 1: $f(\cdot) \rightarrow b$ **means** b is an output of the function f on $\cdot$; $a \leftarrow b$ **means** b is used as input to generate/achieve a
Variables: u: particles output state; q: particles chosen action from input state
1) Initial Explorations:
for $e = 1...N_e$ **do**
  Sample initial state x_0 from gaussian distribution
  for $j = 1...T$ **do**
    Simulate M particle rollouts with policy $\pi_r(x_0) \rightarrow u_j, q_j$
  end for
end for
2) Reinforce Model:
Initialize the GP models
for each GP **do**
  Pretrain the GP $\leftarrow (u, q, k)$
end for
3) Reinforce Policy:
for $j = 1...N_{opt}$ **do**
  Simulate M particle rollouts on $\pi_\theta(x_0, p_d) \rightarrow u, q$
  Compute cost $\hat{J}(\theta_j)(u, q)$ from particles
  Compute gradient $\nabla_\theta \hat{J}(\theta_j)$ through backpropagation;
  Gradient-based policy update $\pi_{\theta_{j+1}}(u|q)$;
  Update monitoring signal s_j;
  if monitoring signal update condition is True **then**
    Update p_d, α_{lr}, and σ_s;
  end if
  if termination condition is False **then**
    break;
  end if
end for
4) Apply Control Policy:
Apply updated policy to the system and collect data;
return trained policy, learned GP model;

One notable constraint in the MC-PILCO framework, as observed by the authors, is the difficulty in handling long episode horizons, particularly those involving hundreds of steps. This challenge originates from the compounding uncertainty, which leads to 'exploding' particle rollouts that obstruct the extraction of substantial policy gradient updates. The authors' use of polynomial kernel components seems to accentuate this problem, limiting their ability to decrease the sampling time and manage more rapid dynamics.

The authors employed independent Gaussian Processes (GPs) to model the dynamics across various state dimensions, without venturing into more advanced Multi-Output models (MOGPs). Future research may seek to address this gap by investigating methods to encapsulate the entire state prediction within a single probabilistic model.

The process of updating policies, as described by the authors, involves meticulous tuning and is prone to local minima. Determining the suitable parameters for the uniform distributions from which they sample the RBF network centers is crucial, as is the adjustment of the dropout and learning rate values and decays.

An ongoing challenge recognized by the authors is the escalating computational cost of GPs across successive trials. They attempted to address this by implementing a Sparse Online Gaussian Process (SOGP) to minimize the data volume. However, this only partially alleviates the issue and further complicates the handling of long horizons. The potential use of Normalizing Flows (NFs) could offer a way to streamline the model, thereby accelerating rollout predictions and policy learning.

The authors recognized that addressing the 'exploding' particles issue could significantly enhance the MC-PILCO algorithm's effectiveness. Consequently, their suggested path forward entails a two-pronged effort: to deepen the understanding of predictive models and to develop more robust policies and update strategies.

The purpose of our paper is to build on this foundation, exploring and addressing these limitations. We aim to bring a fresh perspective and innovative solutions to these challenges, pushing the boundaries of what is currently possible with MC-PILCO.

2.3. Normalizing Flows

Normalizing Flows (NFs) provide a powerful approach to generative modeling, allowing for the construction of complex, multi-modal distributions, while maintaining the ability to perform exact inference and efficient sampling. They achieve this by transforming a simple, easy-to-sample distribution (such as a multivariate Gaussian) into a more complex one, through a series of invertible and differentiable transformations.a

Mathematically, let $z \sim p_z(z)$ be a random variable following a simple distribution (e.g., a multivariate Gaussian), and let f be an invertible transformation, then we can define a new random variable $x = f(z)$. The distribution of x is given by the change of variables formula:

$$p_x(x) = p_z(f^{-1}(x)) \left| \det \frac{\partial f^{-1}}{\partial x} \right| \tag{1}$$

where $\left| \det \frac{\partial f^{-1}}{\partial x} \right|$ is the absolute value of the determinant of the Jacobian of f^{-1} evaluated at x. This formula allows us to evaluate the exact density of any point x in the space of the target distribution.

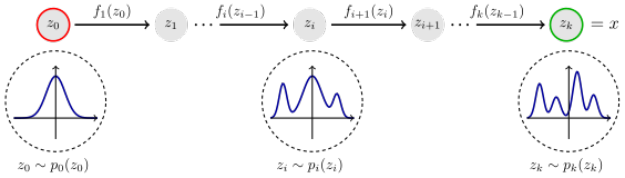

Figure 1. Normalizing Flows transform tractable probability densities to those that are arbitarly complex through a series of bijective mappings

When the transformation f is a composition of multiple simpler transformations, $f = f_1 \circ f_2 \circ \ldots \circ f_K$, the resulting distribution can model more complex structures, making NFs particularly useful for multi-modal distributions.

2.4. Conditional Normalizing Flows

Conditional Normalizing Flows (CNFs) extend NFs by conditioning the transformation on additional inputs. This allows the generated output to be influenced by these inputs, making CNFs useful for a variety of tasks, including generating specific classes of objects or predicting future states in a time-series.

In our work, we condition our flows on the mean μ and variance σ^2 predicted by a Gaussian Process (GP). Let $c = [\mu, \sigma^2]$ be the conditioning variables, the conditional distribution of x given c is given by:

$$p_{x|c}(x|c) = p_z(f^{-1}(x|c)) \left| \det \frac{\partial f^{-1}}{\partial x} \right| \tag{2}$$

where $f^{-1}(x|c)$ is now a conditional transformation.

Adding multiple layers of conditional flows, each parameterized by neural networks, allows us to create highly flexible models. The parameters of these networks are typically learned using maximum likelihood estimation, and can be optimized efficiently using backpropagation and stochastic

gradient descent. This makes CNFs well-suited for a variety of tasks, such as density estimation, variational inference, and generative modeling, where the ability to efficiently sample from complex, multi-modal, and high-dimensional distributions is crucial.

2.5. Temporal Normalizing Flows

Temporal Normalizing Flows (TNFs) further extend NFs by making the transformation time-dependent, allowing it to model time-varying distributions. This approach was proposed in (Both & Kusters, 2019) for scalar normalizing flows, where successive snapshots of an evolving probability distribution $p_x(x,t)$ are mapped to the same tractable latent distribution $p_z(z)$ through a time-dependent transformation $x(t) = T^{-1}(z;t)$. This transformation satisfies the determinant of the Jacobian:

$$\det J_T = \frac{\partial z}{\partial x} \tag{3}$$

Therefore, the normalizing flow between the latent and target density becomes:

$$p_x(x,t) = p_z(z,t) \left| \frac{\partial z}{\partial x} \right| \tag{4}$$

2.6. Real-valued Non-Volume Preserving Flows

Real-valued Non-Volume Preserving (RealNVP) flows are a specific type of normalizing flow which enable simple and efficient computation of both the forward and inverse transformations and their Jacobians. They are based on a simple bijective transformation, which acts differently on different parts of the input space, hence allowing for the modeling of complex, multi-modal distributions.

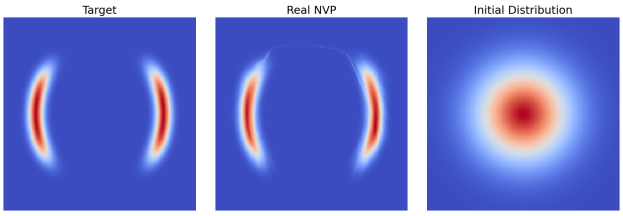

Figure 2. Mapping a Multivariate Gaussian to the Half Moon Distribution with the Real-valued Non-Volume Preserving Flows used in this work. We demonstrate the ability to transform a Gaussian into a complex distribution under conditions

The RealNVP model in our study consists of several transformations, each parameterized by two functions, s and t, which are implemented as neural networks. These functions are responsible for scaling and translation operations, respectively, and allow the model to learn complex transformations.

The transformation for RealNVP is defined as follows:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}) \end{aligned} \tag{5}$$

where $x_{1:d}$ are the dimensions left unchanged, $x_{d+1:D}$ are the dimensions that are transformed, $\odot$ represents element-wise multiplication, and s and t are functions implemented as neural networks.

The Jacobian determinant of this transformation is easy to compute, which facilitates both forward and inverse operations:

$$\det \mathbf{J} = \exp(\sum s(x_{1:d})) \tag{6}$$

Forward operation: The forward operation corresponds to transforming an input x from the base distribution to the target distribution. This is done by applying the transformation iteratively to the input. The log determinant of the Jacobian is computed at each step and summed to give the total log determinant.

Inverse operation: The inverse operation maps an input y from the target distribution back to the base distribution. This is done by applying the inverse of the transformation iteratively. Like in the forward operation, the log determinant of the Jacobian is computed at each step and summed.

Sampling: Sampling from the model involves first sampling a point z from the base distribution, and then applying the inverse operation to map z to the target distribution.

These operations are consistent across all the flows we have explored in this study, including vanilla NF, conditional NF, and temporal NF. The power of these methods comes from the ability to learn complex, multi-modal distributions, and the capability to perform exact density estimation and efficient sampling, which is crucial for the improvement of the dynamics model.

2.7. Experiments

In order to evaluate the performance of the proposed normalizing flows and their applications in model-based reinforcement learning, we conduct several experiments with different setups. A particular emphasis is placed on the importance of multivariate time-varying densities in these settings.

2.7.1. Importance of Multivariate Time-Varying Densities

In the study of model-based reinforcement learning under uncertainty in probabilistical algorithms like MC-PILCO, multivariate time-varying densities provide a rich framework to capture the complex and evolving dynamics of the environment. Reinforcement learning, by its very nature, must deal with an ever-changing landscape of states and outcomes, driven by the agent's actions and the response of the environment. In such a setting, the ability to model and predict the evolution of the system's state, particularly when it is subject to stochastic influences, is of utmost importance. This is where the power of normalizing flows, and in particular their ability to model complex, multi-modal distributions, comes into play (Lu et al., 2022). The following dynamics tests were chosen to explore the capabilities of NF in modelling time-varying distributions, especially chaotic and multi-modal systems.

2.7.2. Experiment 1: Capturing the Evolution of a Unimodal Distribution into a Multimodal One

Our first experiment considers a two-dimensional Stochastic Differential Equation (SDE) with Brownian motion, defined as follows:

$$dX1(t) = (8X1(t) - X1^3(t))dt + dB1(t), \quad (7)$$

$$dX2(t) = (8X2(t) - X2^3(t))dt + dB2(t), \quad (8)$$

where $B1$ and $B2$ are two independent scalar Brownian motions, and the initial conditions $(X1(0), X2(0))$ are drawn from a standard bivariate normal distribution.

Using a training dataset generated with the same number of snapshots, and the same number of samples per snapshot as the example in Eq. (11), we aim to capture the gradual evolution of a unimodal distribution into a multimodal one. The training data is collected from the numerical solution of this SDE with the provided initial conditions, and the normalizing flow is then trained on this data. The results, shown in Fig. 5, demonstrate the capability of our model to capture this complex transformation.

2.7.3. Experiment 2: The Lorenz Attractor

The second experiment involves the Lorenz attractor, a classic system in the study of chaotic dynamics. The Lorenz attractor is defined by the following system of differential equations:

$$\frac{dx}{dt} = \sigma(y - x), \quad (9)$$

$$\frac{dy}{dt} = x(\rho - z) - y, \quad (10)$$

$$\frac{dz}{dt} = xy - \beta z, \quad (11)$$

where σ, ρ, and β are parameters. The Lorenz attractor is a well-characterized example of chaotic dynamics with a rich surrounding literature. By studying this system, we aim to demonstrate the capability of normalizing flows in capturing and predicting the complex, multi-modal behaviors inherent in chaotic systems.

2.7.4. Experiment 3: The Van der Pol Oscillator

Our third experiment studies the Van der Pol oscillator, a non-linear dynamical system given by the following differential equation:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0, \quad (12)$$

where μ is a scalar parameter. For certain values of μ, the system exhibits chaotic behavior. In the stable states, particles oscillate in a predictable manner around a region of stability, forming a limit cycle. The challenge here for the normalizing flow is to model the distribution of particles in space-time within these specific regions, capturing both the stable oscillatory behavior and the transitions between different states.

2.7.5. Experiment 4: Advection-Diffusion Random Walk

The fourth experiment is an advection-diffusion random walk in one dimension. This simple experiment allows us to understand what happens under random processes and makes it easy to manipulate the parameters and run multiple divergence experiments. The Python code to generate the walk is as follows:

This model simulates the random motion of particles subjected to advection and diffusion, allowing us to study how well normalizing flows can model the evolution of the system's state under stochastic influences.

2.7.6. Experiment 5: The Cartpole Environment with Gaussian Process and Conditional Normalizing Flows

The final experiment involves the Cartpole environment, a well-established benchmark in reinforcement learning. The Cartpole is an unstable, control-oriented system where a

pole is attached to a cart moving along a frictionless track, and the aim is to balance the pole upright by only moving the cart horizontally.

In our experiment, we introduced heavy noise into the environment to mimic real-world uncertainties. To handle this noisy environment, we applied a multivariate Gaussian Process (GP) with a squared exponential kernel, trained using exact marginal log likelihood. The GP provides a probabilistic model of the Cartpole environment, giving both mean predictions and associated uncertainties.

Following the GP, we utilized Conditional Normalizing Flows (CNF) to propose new particle distributions that more accurately reflect the dynamics of the noisy environment. As discussed earlier, the CNF is conditioned on the first two moments predicted by the GP, thus integrating the uncertainty information into the model. This is formalized in the CNF equation presented in the previous sections.

The aim of this experiment is to demonstrate how CNFs can enhance model-based reinforcement learning under uncertainty by providing a more accurate and efficient representation of the environment's state distribution.

3. MC-PILCO-CNF

Here we present the modified MC-PILCO algorithm with a conditional normalizing flow added into the dynamics model. GP predictions are made about the next step, the mean and variance are passed to the CNF which transforms the Gaussian into a complex distribution. Particles are then sampled from the CNF distribution, the cost estimations are conducted and the Policy is reinforced. We adapted the differentible particle filter outlined by (Chen et al., 2021) by using their proposed CNF.

For the loss function of the CNF NN, we chose negative log likelihood, which is a function of the prior distribution ω, which we chose to be multivariate normal, and the jacobian output from passing c_j through the inverse of the flow.

Additionally, our CNF NN consisted of 2 sequences of the FCNN network. FCNN has the following architecture: Linear, Tanh, Linear, Tanh, Linear. Importantly, the hidden size of the Linear layers must be the length of the context vector c_j fed into the flows for the dimensions to work out.

Following is the algorithm employed by MC-PILCO-CNF:

Algorithm 2 MC-PILCO-CNF incorporates flow-based neural networks to improve the MC-PILCO algorithm. It includes additional parameters for the number of explorations, number of flow epochs, the flow neural network, and the flow prior distribution. The algorithm consists of initial explorations, reinforcing the model and flows, reinforcing the policy, and applying the control policy, similar to MC-

Algorithm 2 MC-PILCO-CNF

Input: In addition to parameters described in MC-PILCO, we have: number of expxlorations N_e, number of flow epochs N_f, flow NN Ω, flow prior distribution ω.
Notation Remark 1: $f(\cdot) \rightarrow b$ **means** b is an output of the function f on $\cdot$; $a \leftarrow b$ **means** b is used as input to generate/achieve a
Variables: u: particles output state; m: particles output state mean, v: particles output state variance, o: particles input state observations, q: particles chosen action from input state; $\phi = u, m, v, o, s$.
1) Initial Explorations:
for $e = 1...N_e$ **do**
  Sample initial state x_0 from gaussian distribution
  for $j = 1...T$ **do**
    Simulate M particle rollouts with policy $\pi_r(x_0) \rightarrow \phi_j$
    Get the context vector $c_j \leftarrow (m_j|v_j|o_j|s_j) \leftarrow \phi_j$
    Get the particle updates through the flow's inverse $\Omega_{inv}(c_j, u_j) \rightarrow c_{j,update}$
  end for
end for
2) Reinforce Model (same as MC-PILCO)
3) Reinforce Flows Model:
for $i = 1...N_f$ **do**
  for $j = 1...T$ **do**
    Get the context vector $c_j \leftarrow (m_j|v_j|o_j|s_j) \leftarrow \phi_j$
    Get the particle updates through the flow's inverse $\Omega_{inv}(c_j, u_j) \rightarrow c_{j,update}$
    Gradient based flow NN parameter update $\leftarrow (c_{j,update}, \omega)$
  end for
end for
4) Reinforce Policy:
for $j = 1...N_{opt}$ **do**
  Simulate M particle rollouts on $\pi_\theta(x_0, p_d) \rightarrow \phi$
  Compute cost, gradient, update policy (same as MC-PILCO)
  Train flows (same as reinforce flows but with *no_grad* off) $\leftarrow \phi$
  Update parameters $\leftarrow \sigma_j$ (same as MC-PILCO)
end for
5) Apply Control Policy (same as MC-PILCO)

PILCO.

4. Results

We demonstrated the potential of our dynamics model to learn complex multi-modal distributions in a variety of circumstances.

In Fig. 13, we display the capacity of our model to map rollout trajectories under the constraints of physics. Moreover, we simulated scenarios in which rollouts diverged into two stable states as shown in Fig. 11. However, the limitation of temporal flow precluded us from sampling and performing inverse operations from more than one realNVP flow layer.

Our model also demonstrated the capability to model the behavior of the Vander Pol oscillator, as shown in Fig. 10. This was a challenging case since particles from a normal distribution fall into the oscillator's attractor and rotate around it, forming a cyclical but changing distribution. While our model was able to semi-accurately model this cyclical distribution, further work in this area is warranted.

Table 1. Values for cartpole policy optimization parameters.

PARAMETER	DESCRIPTION	VALUE
p_d	DROPOUT PROBABILITY	0.25
Δp_d	p_d REDUCTION COEFFICIENT	0.125
α_{lr}	ADAM STEP SIZE	0.01
$\alpha_{lr_{min}}$	MINIMUM STEP SIZE	0.0025
α_s	EMA FILTER COEFFICIENT	0.99
σ_s	MONITORING SIGNAL THRESHOLD	0.08
n_s	NUM ITERATIONS MONITORING	200
λ_s	σ_s REDUCTION COEFFICIENT	0.5
M	NUMBER OF PARTICLES	400
t	NUMBER OF TRIALS	5
N_{opt}	NUMBER OF OPTIMIZATION STEPS	1000

Table 2. Values for cartpole policy optimization parameters.

PARAMETER	DESCRIPTION	VALUE
N_f	NUMBER OF FLOW EPOCHS	100
α_{lr_f}	ADAM LR FOR FLOW NN	0.001

Fig. 9 demonstrates the particles' distribution stabilizing to specific positions in the state space over time, forming a multi-modal distribution. We found that our model effectively represented this distribution. It is interesting to observe the traces between regions of density. Again, our flow model was limited to a single flow layer.

Fig. 16 demonstrates the potential of a Gaussian Process (GP) to predict uncertainty dynamics, providing an understandable model for these phenomena. This serves as a baseline for understanding the predictive capabilities of our model. In contrast, Fig. 17 showcases the improvement of our method over the GP baseline. The model shows promise in learning the dynamics and converging, but a more comprehensive exploration of the model's specific strengths is required. These results indicate that conditioning on the GP output can be an effective method, though more tests need to be conducted to confirm this.

As for the more complex case of the Lorenz system, Fig. 14 illustrates the evolution of the Lorenz attractor. However, as shown in Fig. 15, the Temporal Normalizing Flow (TNF) struggles to fully capture the dynamics of this system. The model was only partially successful in predicting the dynamics, even when the hidden state size was increased. The decreasing particle density over time may suggest the presence of exploding particles within the system, which is an area that would benefit from further investigation.

Future work should investigate the data-efficiency of CNF. The field lacks research considering temporal CNFs. Moreover, physics-based RL, which improves sample efficiency, and Physics-Informed Neural Networks (PINNs), which demonstrate state-of-the-art extrapolation, are important areas of study. The use of CNF in policy exploration and deconstructing the Gaussian assumptions in Model-Based Reinforcement Learning (MBRL) will likely prove to be interesting and promising directions.

We attempted to first get baseline results of cartpole on MC-PILCO. The following parameters yielded the best result shown in the figure (also below).

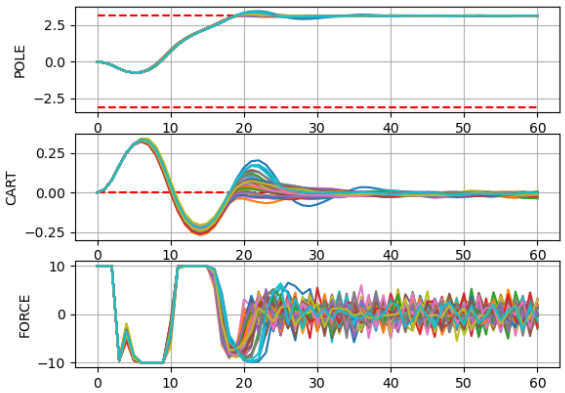

Figure 3. Sampled states of cartpole over policy iteration. The x axis represents the policy iteration number. The Pole subplot represents the first set of sampled states, the cart subplot represents the second set of sampled states, and force subplot represents the chosen action

We attempted to integrate CNF as outlined in our algorithm. Since the time to run would have been insufficient given constraints, we decided to reduce the number of trials from $t = 5$ to $t = 1$. However, although we were able to get the cost to reduce, we were never able to keep the pole steady for longer than 30 rollout time units. The following diagram shows our best attempt of MC-PILCO-CNF:

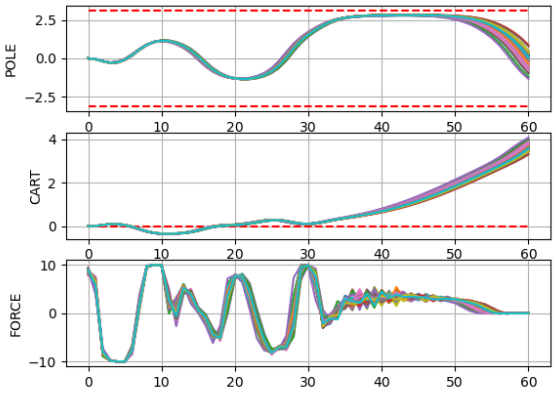

Figure 4. Sampled states of cartpole over policy iteration for MC-PILCO-CNF. Each subplot represents the values of each dimension of the state space over many simulated rollouts

In addition to the values for cartpole policy optimization used by MC-PILCO (with the exception that $t = 1$), we used these normalizing flow parameters:

We discovered a potential reason for the lack of control. If we reduce the number of trials in MC-PILCO (the baseline) to 1, we get a similar result, where the cart does not properly stabilize:

With both the CNF and baseline results at $t = 1$, it was fair to compare them.

In the baseline, the cart starts moving at rollout time unit 22 as opposed to 30 with CNF enabled. Additionally, the quick and successive alternation of force is more nuanced in the CNF plot as apposed to the baseline plot. In fact, in the baseline, the force is 0 for the last 25 time rollout units. Both findings point towards the benfit of using normalizing flows.

Unfortunately, there are some unresolved questions. For instance, in the cartpole rollout expirments 7, for the baseline result, the pole seems to be stabilizing again at 60 rollout units whereas it is falling steeply when CNF is introduced. Interestingly, with CNF, the pole's movements as seen in the subplots are more delayed and 'sluggish' compared to the baseline.

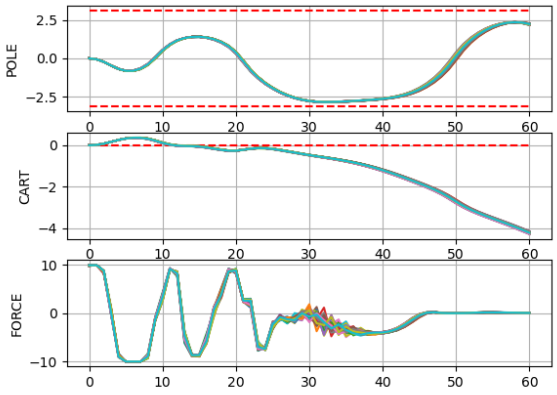

Figure 5. Sampled states of cartpole over policy iteration for MC-PILCO when $t = 1$.

Since we had two signals suggesting CNFs improved MC-PILCO, and one signal suggesting it did not, we decided to see how quickly cost convergence occured in each model.

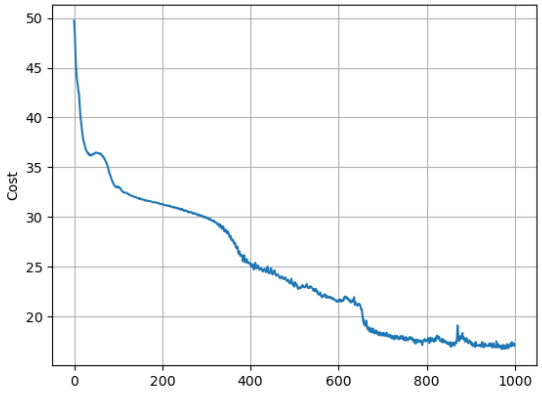

Figure 6. Cost over optimization steps for MC-PILCO with $t = 1$.

As can be seen, the cost achieved with CNF is about 10 points lower than that from the baseline.

5. Conclusion

Our series of experiments and analyses have provided considerable insight into the capabilities and limitations of multimodal distributions in the realm of state exploration and sampling. The use of multimodal distributions, as opposed to unimodal ones, has shown to be particularly beneficial in the context of model-based reinforcement learning (MBRL).

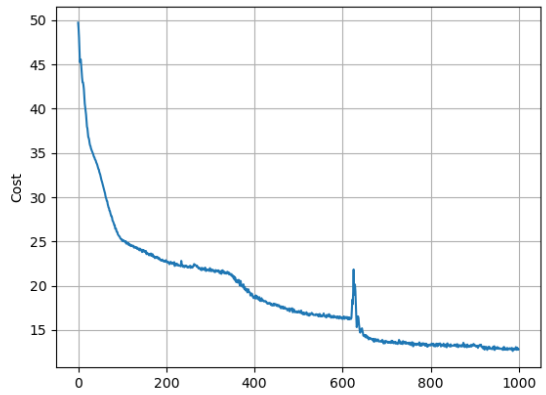

Figure 7. Costs over optimization steps for MC-PILCO-CNF with $t = 1$.

This is especially true when considering the need for stochastic policies and exploration in MBRL.

Stochastic policies, which introduce randomness into the decision-making process, can be crucial for effective exploration in MBRL. They allow the model to explore different parts of the state space, which can lead to better overall performance. Normalizing flows, with their ability to model complex, multimodal distributions, can potentially enhance this process. They can provide a more accurate and nuanced representation of the state space, allowing for more effective exploration.

In contrast to deterministic dynamics models, which can only represent a single outcome for each state-action pair, models that can perform density estimation, such as those using normalizing flows, can represent a range of possible outcomes. This can be particularly important in complex environments where the same action can lead to different outcomes depending on the context.

Multimodal distributions are also necessary in many robotics and engineering problems. For instance, in control problems, there may be multiple viable control strategies for a given state, each corresponding to a different mode of the distribution. Similarly, in robotics, a robot may need to choose between multiple paths to reach its destination, each path corresponding to a different mode. Our model's ability to represent these multimodal distributions can therefore be highly beneficial in these contexts.

Our results have shown that the use of normalizing flows can lead to significant improvements in cost convergence, as evidenced in our MC-PILCO-CNF experiments. However, these improvements come at the cost of increased training time, which is a significant trade-off to consider. In our study, we incorporated the two modes of the Gaussian process into our CNF dynamics model. However, this approach combines the computational complexity of the Gaussian process and the normalizing flow, which poses a challenge. To overcome this bottleneck, it may be advantageous to replace the Gaussian process entirely with the flow.

In the context of the cartpole environment, Deep-PILCO (Chua et al., 2018) demonstrated reduced data efficiency when using a neural network dynamics model. Therefore, future research with CNF dynamics models should establish appropriate constraints for the CNF and incorporate suitable inductive bias. The recent developments in physics-informed CNF have sparked our interest in developing sample-efficient dynamics models, and we believe this direction holds promise.

Looking forward, it would be interesting to explore more chaotic systems, where the benefits of using a dynamics model may be more pronounced. While Gaussian models may suffice in many mechanical systems, there are likely scenarios where a more complex model would be beneficial. Implementing CNF into other aspects of the algorithm, such as policy optimization, could also be a fruitful direction for future research.

Future work should also investigate the potential benefits of using other types of normalizing flows, such as Glow, which has shown promising results in image-based flows. It would also be worthwhile to explore alternative base neural networks beyond FCNN, such as Planar or Radial flows.

In conclusion, our work has highlighted the potential benefits of using normalizing flows in MBRL, particularly in the context of state exploration and sampling. However, further research is needed to fully understand these benefits and to explore potential improvements.

References

Amadio, F., Libera, A. D., Antonello, R., Nikovski, D., Carli, R., and Romeres, D. Model-Based Policy Search Using Monte Carlo Gradient Estimation with Real Systems Application. *IEEE Transactions on Robotics*, 38 (6):3879–3898, December 2022. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2022.3184837. URL http://arxiv.org/abs/2101.12115. arXiv:2101.12115 [cs].

Both, G.-J. and Kusters, R. Temporal Normalizing Flows, December 2019. URL http://arxiv.org/abs/1912.09092. arXiv:1912.09092 [physics, stat].

Chen, X., Wen, H., and Li, Y. Differentiable Particle Filters through Conditional Normalizing Flow, Novem-

ber 2021. URL http://arxiv.org/abs/2107.00488. arXiv:2107.00488 [cs].

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models, November 2018. URL http://arxiv.org/abs/1805.12114. arXiv:1805.12114 [cs, stat].

Gal, Y., McAllister, R. T., and Rasmussen, C. E. Improving PILCO with Bayesian Neural Network Dynamics Models.

Klink, P. Model-Based Reinforcement Learning from PILCO to PETS. In Belousov, B., Abdulsamad, H., Klink, P., Parisi, S., and Peters, J. (eds.), *Reinforcement Learning Algorithms: Analysis and Applications*, Studies in Computational Intelligence, pp. 165–175. Springer International Publishing, Cham, 2021. ISBN 978-3-030-41188-6. doi: 10.1007/978-3-030-41188-6_14. URL https://doi.org/10.1007/978-3-030-41188-6_14.

Lu, Y., Maulik, R., Gao, T., Dietrich, F., Kevrekidis, I. G., and Duan, J. Learning the temporal evolution of multivariate densities via normalizing flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3): 033121, March 2022. ISSN 1054-1500, 1089-7682. doi: 10.1063/5.0065093. URL http://arxiv.org/abs/2107.13735. arXiv:2107.13735 [cs, math, stat].

Ward, P. N., Smofsky, A., and Bose, A. J. Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies, June 2019. URL http://arxiv.org/abs/1906.02771. arXiv:1906.02771 [cs, stat].

A. Appendix.

Additional plots exploring our NF / CNF / TNF dynamics models

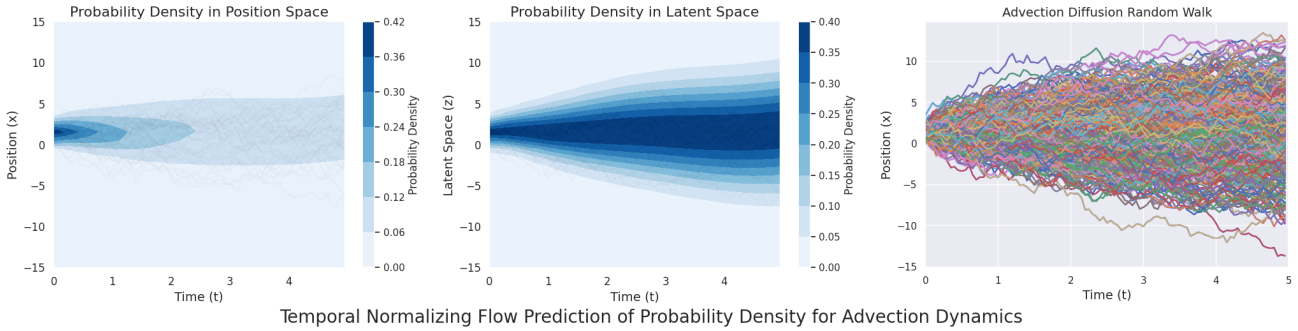

Figure 8. This plot presents the Temporal Normalizing Flow Prediction of Probability Density for Multi-Modal Advection Dynamics. It comprises three subplots: the first two illustrate the probability density in position and latent spaces, respectively, while the third shows the progression of an Advection Diffusion Random Walk. Together, these plots depict the dynamics of the system and the multi-modal nature of its state distributions over time.

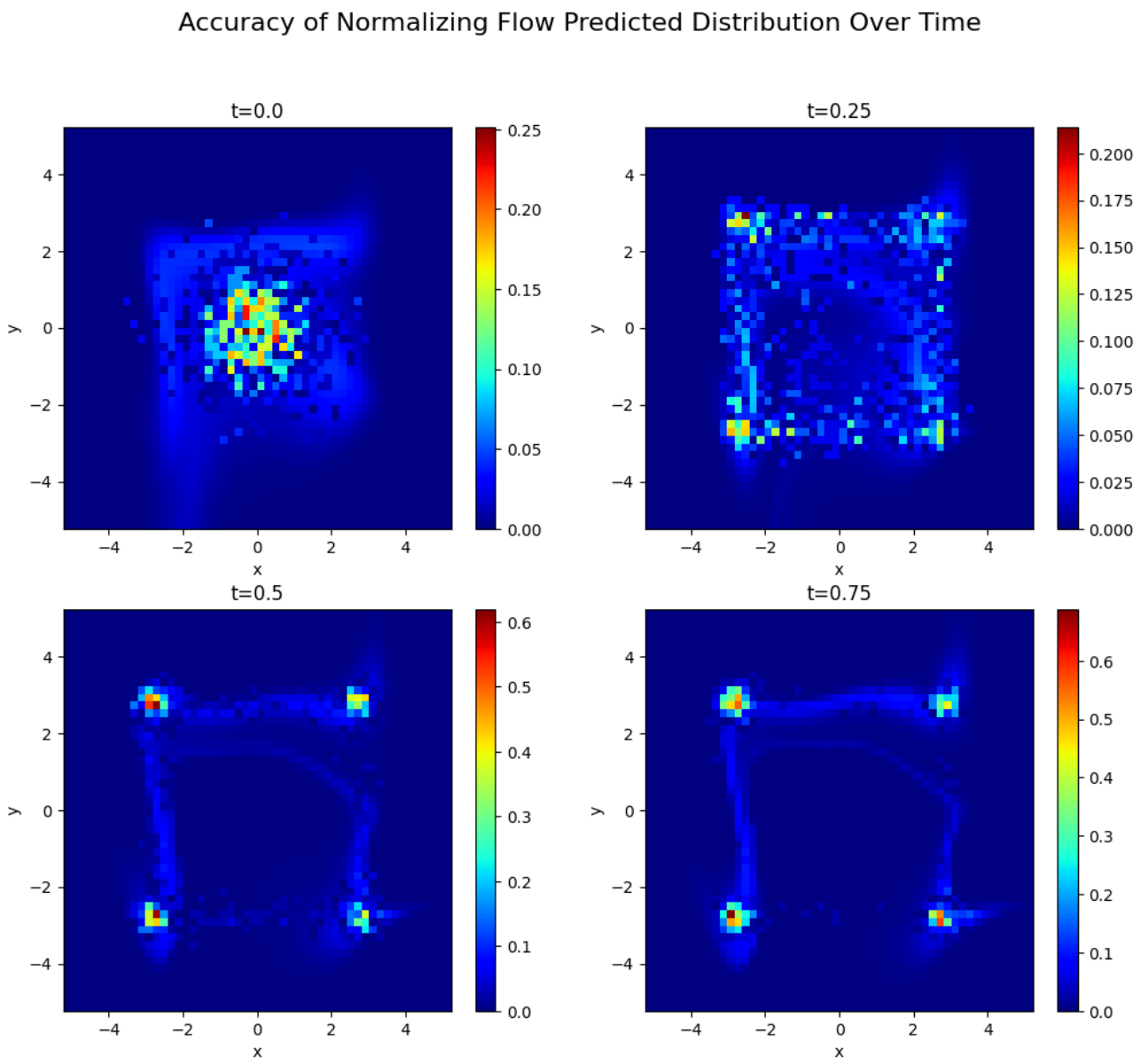

Figure 9. This plot presents the Temporal Normalizing Flow Prediction of Probability Density for Multi-Modal Advection Dynamics. It comprises three subplots: the first two illustrate the probability density in position and latent spaces, respectively, while the third shows the progression of an Advection Diffusion Random Walk. Together, these plots depict the dynamics of the system and the multi-modal nature of its state distributions over time.

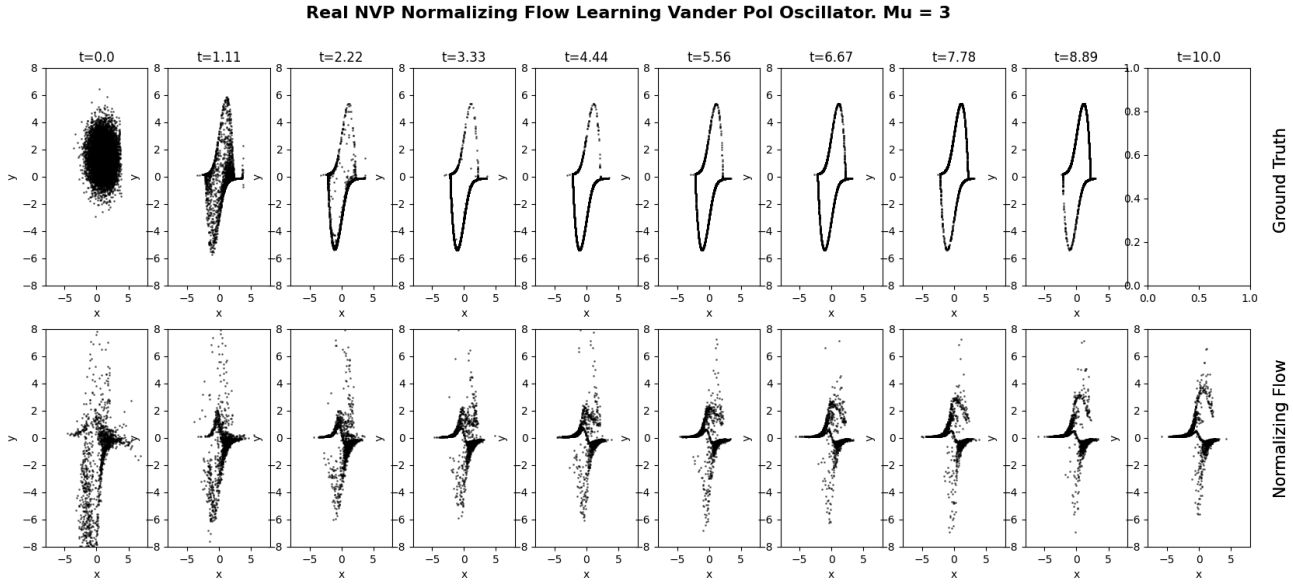

Figure 10. Temporal Normalizing Flow Learning the Vander Pol Oscillator Function

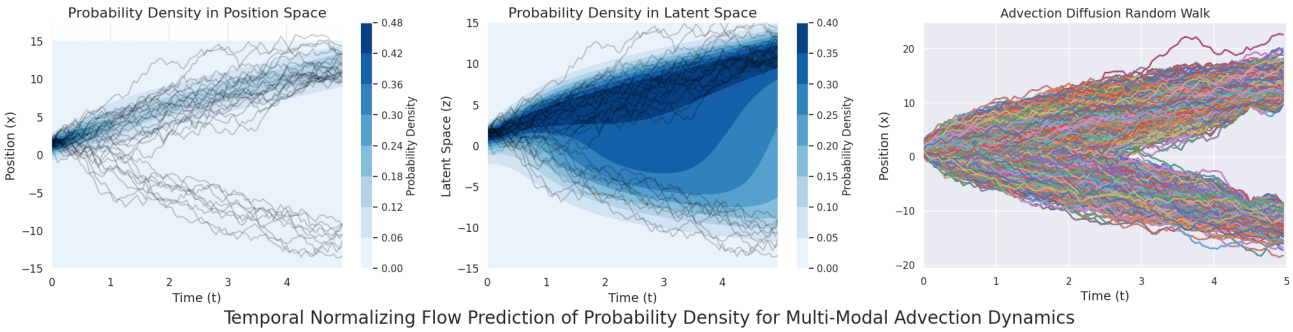

Figure 11. Diverging random walks simulating situations where an agents state space will converge to two highly likely future states

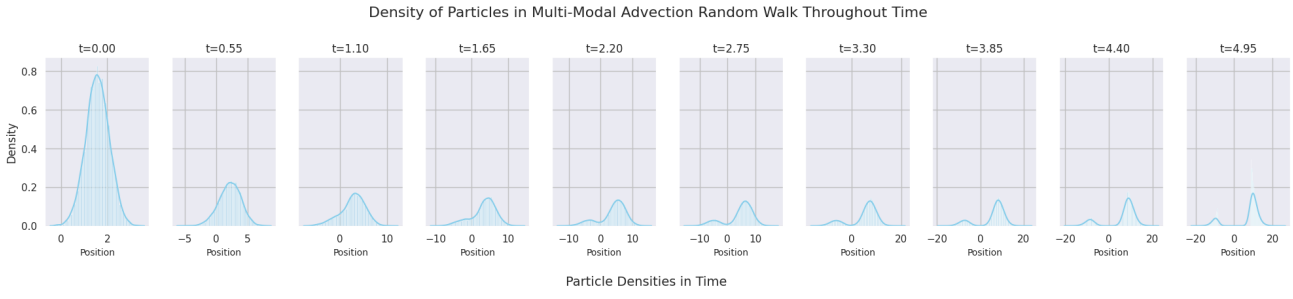

Figure 12. Particles simulated into the future form a bimodal distribution

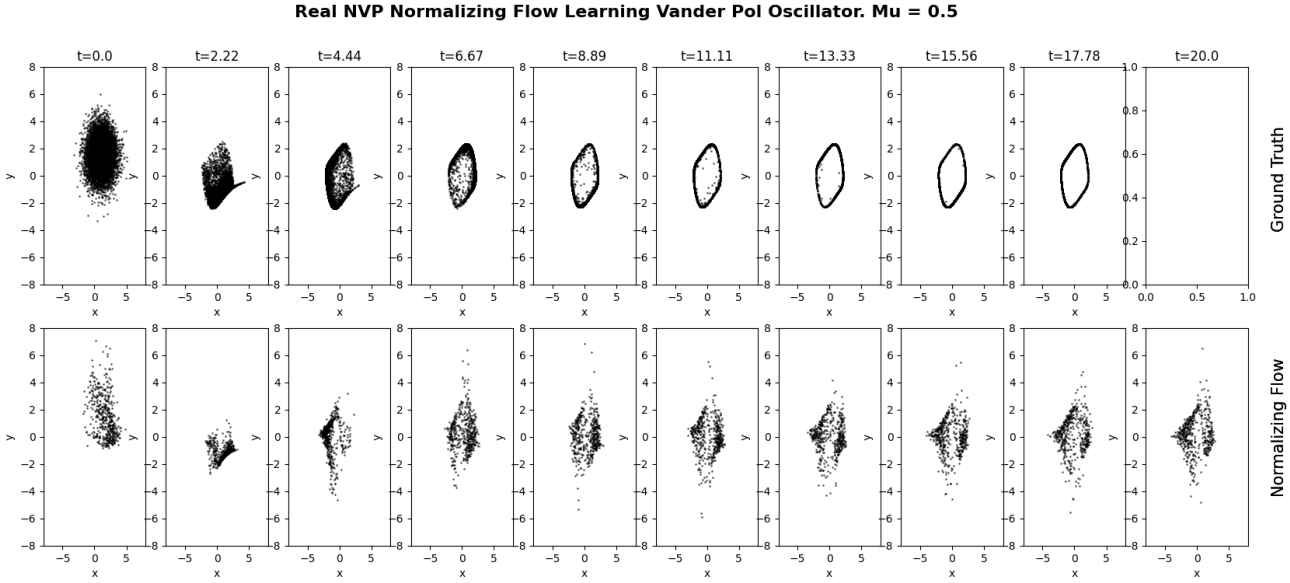

Figure 13. Temporal Normalizing Flow with Real NVP learning the cyclic probability distribution of the Vander Pol Oscillator (Mu = 0.5)

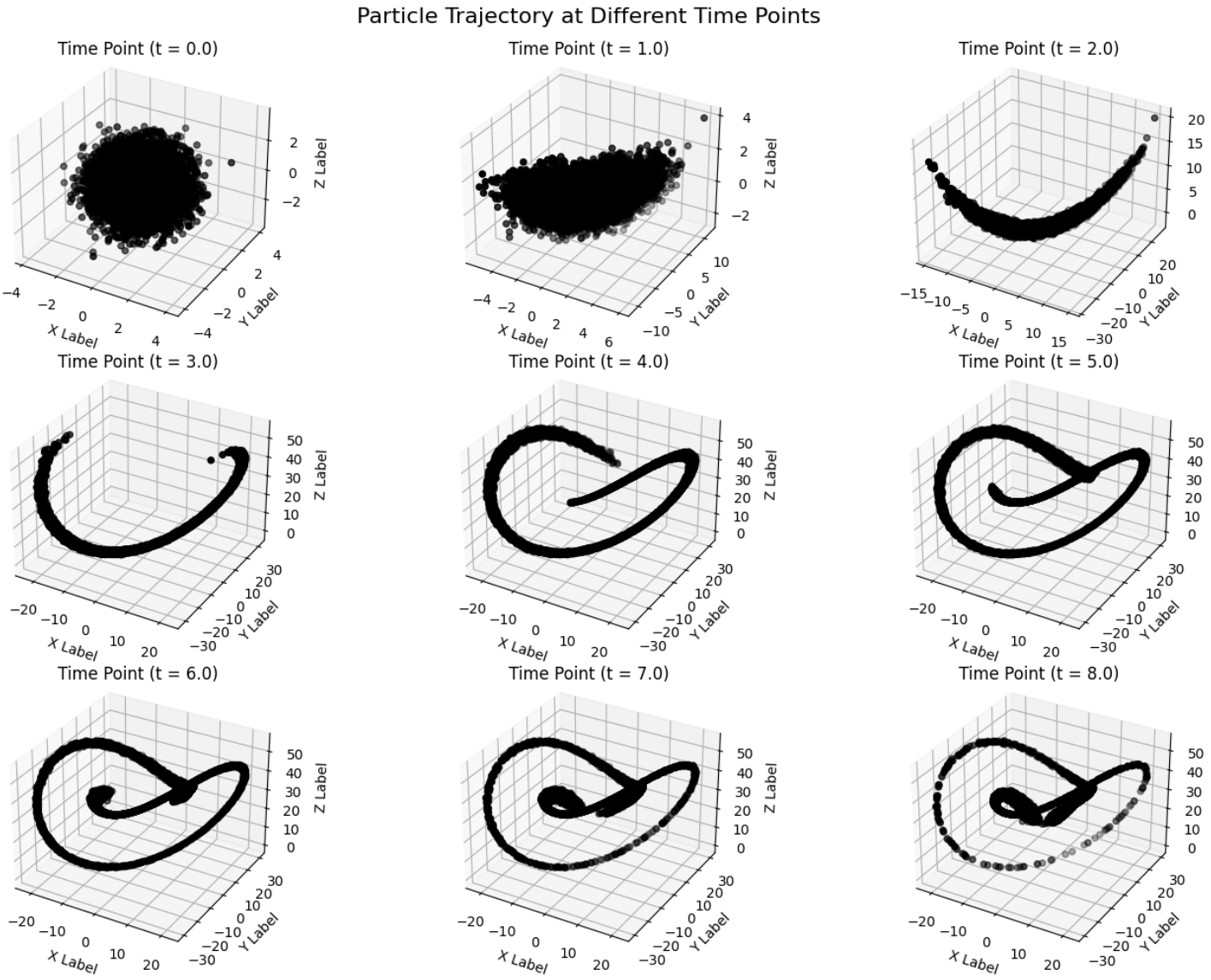

Figure 14. Evolution of The Lorenz System

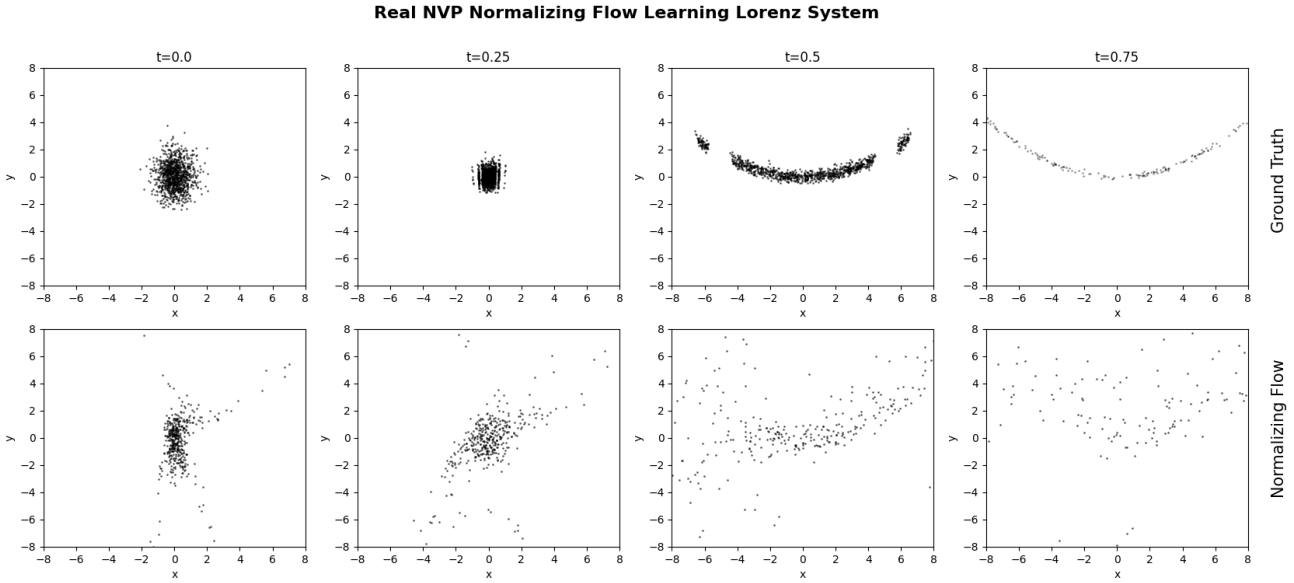

Figure 15. Evolution of The Lorenz System under the temporal normalizing flow. The plot represents a small subsection of the attractor. While only one flow layer, the model is not completely capable of predicting the dynamics.

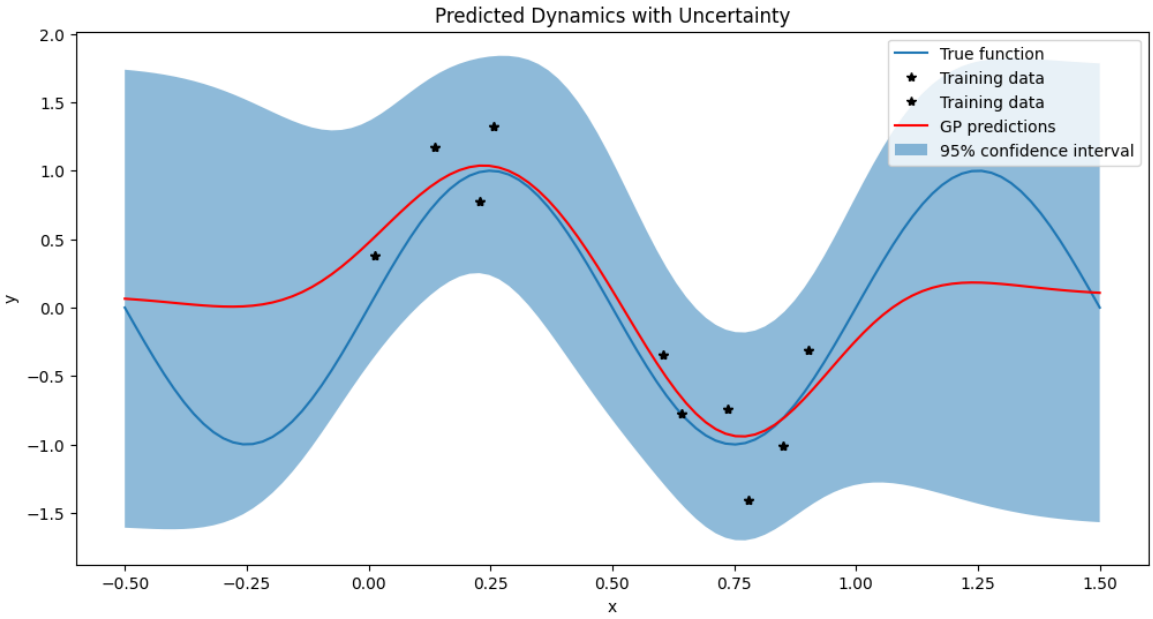

Figure 16. Predicting the uncertainty of data-sparse Cartpole environment with a Gaussian Proccess using Squared Exponential Kernel

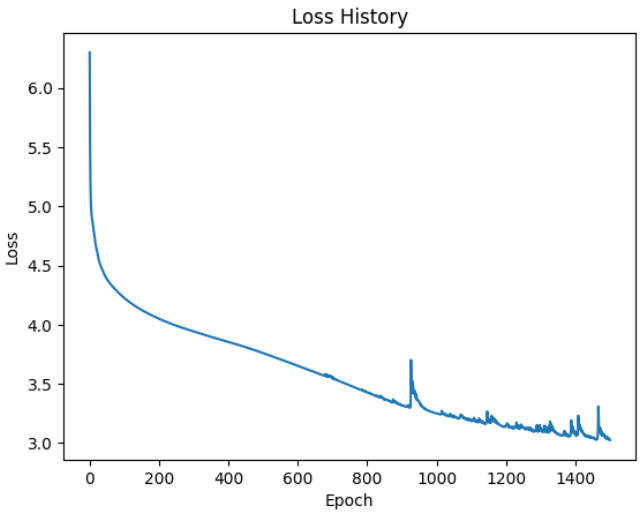

Figure 17. Loss history for CNF, conditioned on the output of the Gaussian predictions.