Learning Density Distribution of Reachable States for Autonomous Systems

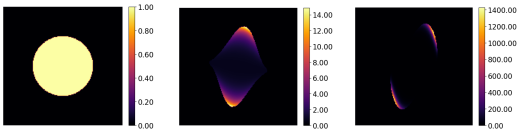
Anonymous Author(s)
Affiliation
Address
email

Abstract: State density distribution, in contrast to worst-case reachability, can be leveraged for safety-related problems to better quantify the likelihood of the risk for potentially hazardous situations. In this work, we propose a data-driven method to compute the density distribution of reachable states for nonlinear and even black-box systems. Our semi-supervised approach learns system dynamics and the state density jointly from trajectory data, guided by the fact that the state density evolution follows the Liouville partial differential equation. With the help of neural network reachability tools, our approach can estimate the set of all possible future states as well as their density. Moreover, we could perform online safety verification with probability ranges for unsafe behaviors to occur. We use an extensive set of experiments to show that our learned solution can produce a much more accurate estimate on density distribution, and can quantify risks more accurately and flexibly comparing with worst-case analysis.

Keywords: Reachability Density Distribution, Learning Density Distribution, Liouville Theorem

1 Introduction

Reachability analysis has been a central topic and the key for the verification of safety-critical autonomous systems. The majority of existing reachability approaches either compute worst-case reachable sets [1, 2, 3, 4, 5, 6], or use Monte Carlo simulation to estimate the reachable states with probabilistic guarantees [7, 8, 9, 2]. There are several obvious disadvantages of such methods. 1). Worst-case methods often over-approximate the reachable sets and produce very conservative results (due to convex set representations [10, 11, 12, 13], wrap effects [14, 15], low-order approximation [3, 16], etc.), not to say they are usually very computational expensive (also known as the curse of dimensionality). 2). Existing methods do not care about reachable state concentration and produce the same reachable state estimations for different distributions of initial states if those distributions share the same support [1]. That is, current approaches do not compute which states are more likely to be reached. For example, in Fig. 1, the state density distribution of a Van der Pol oscillator evolves over time and concentrates on certain states (the highlighted part in the figure), even if the initial states are uniformly distributed. If one uses an existing reachability algorithm, most likely the results will show that almost the entire space in the figure is reachable, as those methods use worst-case analysis and convex sets to represent the reachable sets.



(a) $t = 0s$. (b) $t = 0.35s$. (c) $t = 0.80s$.

Figure 1: Reachability density of Van der Pol.

In this paper, we aim to tackle the above problems and propose a learning-based method that can compute the density distribution of the reachable states from given initial distributions. The state

[1] The probabilistic guarantees of the sampling-based methods do rely on the form of the initial states distribution. However, the final reachable sets estimate is the same for different distributions with the same support.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

density is much more powerful than worst-case reachability and can better quantify risks. Our proposed method is based on the Liouville theorem [17, 18, 19], which is from classical Hamiltonian mechanics and asserts that the state density distribution function (which is the measurement of state concentration) is constant along the trajectories of the system. Given an autonomous system $\dot{x} = f(x)$ that is locally Lipschitz continuous, the evolution of the state density $\rho(x, t)$ (i.e. the density of state x at time t) is characterized by a Liouville partial differential equation (PDE). We learn the state density as a neural network (NN) while respecting the laws of physics described by the nonlinear Liouville PDE, in a way that is similar to the physics-informed NN [20].

Furthermore, we make two major improvements to make the learned density NN suitable for verification. 1). Instead of learning ρ for a fixed initial distribution $\rho(x_0, 0)$, we learn the density concentration function which specifies the multiplicative change of density from any $\rho(x_0, 0)$. 2). We use a single NN to jointly learn both the reachable state $\Phi(x_0, t)$ and its corresponding density. Moreover, we use Reachable Polyhedral Marching (RPM) [21]—an exact ReLU NN reachability tool—to parse our learned NN as linear mappings from input polyhedrons to output polyhedrons. Using such parsed polyhedrons, we can perform online forward and backward reachability analysis and get the range of density bounds $[\rho_{\min}, \rho_{\max}]$ for each output polyhedron. Together, our method can perform online safety verification by computing the probability of safety (instead of a single yes or no answer) under various initial conditions, even if the system dynamics f is unknown (i.e. the system is a black-box and we only have access to a simulator of $x(t)$). In this way, it also has the potential to collect environmental information in run-time and update its distribution for online safety verification.

We conduct experiments on 10 different benchmarks covering systems from low dimension academic examples to high dimension black-box simulators equipped with either hand-crafted or NN controllers. Surprisingly, without using any ground-truth density data in the learning process, our approach can achieve up to 99.89% of error reduction in KL divergence with respect to the ground-truth value, when comparing to the kernel density estimation method. Moreover, our learned density concentration function can also be used for reachability distribution analysis. We show that in several systems, more than 90% of the states can actually just reside in a small region (less than 10% of the volume of the convex hull for those states) in the state space, which also points out the conservativeness of the worst-case reachability analysis methods in terms of quantifying risks. We also show that different initial distributions can lead to a drastic change in the safety probability, which can help in cases when unsafe is inevitable but can be designed to happen with a very low probability.

Our major contributions are: (1) we are the first to characterize and compute the reachability density/probability distribution for safety analysis, (2) we propose the first data-driven method to learn the state density evolution from arbitrary initial state distributions and give accurate state density estimation (3) we use a variety of examples to show the necessity to perform reachability distribution analysis instead of pure worst-case reachability, to flexibly and accurately quantify safety risks.

Related work Reachability analysis, especially worst-case reachability using set propagation or sampling-based methods, has been studied for decades. The literature on reachability has been extensively studied in many survey reports [1, 6, 22, 23]. In this paper, we only discuss a few works that are closely related to ours.

Hamilton Jacobian PDE has been used to derive the exact reachable sets in [24, 1, 5]. However, the HJ-PDE does not provide the density information. Many data-driven approaches can compute probabilistic reachable sets using scenario optimization [7, 25], convex shapes [8, 9, 26], support vector machines [27, 28], kernel embedding [29], active learning [30], and Gaussian process [2]. However, the probabilistic guarantees they provide are usually in the form that $\mathbf{Prob}(x(t) \in \texttt{estimated set}) > 1 - \epsilon$ with enough samples, instead of the state density distribution. The closest to ours is [31] where Perron-Frobenius and Koopman operators are learned from samples of trajectories. Then, the learned operators can be used to transform the moments of the distribution over time. The distribution at time t is then recovered from the transformed moments. However, as they use moments up to a specific order to represent a distribution, even if the learned operators are perfect, the estimation error of the distribution might not be zero. Also, such a moment-based method is hard to scale to large-dimensional systems. Recently, there is also a growing interest in studying the (worst-case) reachability of NN [32, 33, 34, 35, 21] or systems with NN controllers [36, 4, 37, 38, 39]. In this paper, we use [21] to parse our learned NN as a set of linear

mappings between polyhedrons for online reachability computation, but this step can be replaced with other NN reachability tools.

To measure the probability distribution in the reachable sets, the most naïve approach is to use histograms or kernel density estimation [19], similar to the Monte-Carlo method used in dispersion analysis [40]. But this could lead to poor accuracy and computational scalability [41]. Another approach propagates the uncertainty by approximating the solution of the probability density function (PDF) transport equation [42], which could still be time-consuming due to the optimization process performed at each time step. Our approach finds the PDF transport equation by solving the Liouville PDE [2] using NN. Similar ideas have been explored in [18]. However, the density NN in [18] was learned solely for a fixed initial states distribution and therefore, cannot be used for online prediction. Instead, we jointly learn the reachable states and density changes, and can perform online reachability density computation for any initial states distribution.

The idea of using deep learning to solve PDEs can be traced back to the 1990s [43, 44, 45], where the solutions of the PDE on a priori fixed mesh are approximated by NN. Recently, there is a growing interest in the related sub-fields including: mesh-free methods in strong form [20, 46, 47] and weak form [48, 49], solving high-dimension PDEs via BSDE method [50], solving parametric PDE [51, 52] and stochastic differential equations [53, 54], learning differential operators[55, 56, 57], and developing more advanced toolboxes [58, 59, 60]. Our idea for solving Liouville PDE along trajectories is similar to [50] but without the stochastic term.

2 Preliminaries

We denote by $\mathbb{R}$, $\mathbb{R}^{\geq 0}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times n}$ the sets of real numbers, non-negative real numbers, n-dimensional real vectors, and $n \times n$ real matrices. We consider autonomous dynamical systems of the form $\dot{x}(t) = f(x(t))$, where for all $t \in \mathbb{R}$, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^d$ is the state and $\mathcal{X}$ is a compact set that represents the state space. We assume that $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ is locally Lipschitz continuous. The solution of the above differential equation exists and is unique for a given initial condition x_0. We define the flow map $\Phi : \mathcal{X} \times \mathbb{R} \mapsto \mathcal{X}$ such that $\Phi(x_0, t)$ (also written as $x(t)$ for brevity) is the state at time t starting from x_0 at time 0. Note that system parameters can be easily incorporated as additional state variables with time derivative to be 0.

We analyze the evolution of the dynamical system by equipping it with a density function $\rho : \mathcal{X} \times \mathbb{R} \to \mathbb{R}^{\geq 0}$, which measures how states distribute in the state space at a specific time instant. A larger density $\rho(x, t)$ means the state is more likely to reside around x at time t, and vice versa. The density function is completely determined by the underlying dynamics (i.e., function f) and the initial density map $\rho_0 : \mathcal{X} \mapsto \mathbb{R}^{\geq 0}$. Specifically, given a ρ_0, the density function ρ solves the following boundary value problem of the Liouville PDE [17]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \cdot f) = 0, \quad \rho(x, 0) = \rho_0(x), \tag{1}$$

where $\nabla \cdot (\rho \cdot f) = \sum_{i=1}^{d} \frac{\partial [\rho \cdot f]_i}{\partial [x]_i}$ is the divergence for the vector field $\rho \cdot f$, and $[\cdot]_i$ takes the i-th coordinate of a vector.[3] Intuitively, as shown in [17], Liouville PDE is analogous to the mass conservation in fluid mechanics, where the change of density $\frac{\partial \rho}{\partial t}$ at one point is balanced by the total flux traversing the surface of a small volume surrounding that point. It is hard to solve the Liouville PDE for a closed from of the density function ρ. However, it is relatively easy to evaluate the density along a trajectory of the system. To do that, we first convert the PDE into an ODE as follows. Considering a trajectory $\Phi(x_0, t)$, the density along this trajectory is an univariate function of t, i.e., $\rho(t) = \rho(\Phi(x_0, t), t)$. If we consider the augmented system with states $[x, \rho]$, from Eq. (1) we can easily get the dynamics of the augmented system [19]:

$$\begin{bmatrix} \dot{x} \\ \dot{\rho} \end{bmatrix} = \begin{bmatrix} f(x) \\ -\nabla \cdot f(x)\rho \end{bmatrix}. \tag{2}$$

[2] In the absence of process noise, this PDF transport equation reduces to the stochastic Liouville equation [17]

[3] A general form of Liouville PDE can have a non-zero term on the right-hand side indicating how many (new) states appear or exit from the system during the run time [19]. In all the systems we discuss here, there is no state entering (other than the initial states) or leaving the system, so the right-hand side of Eq. (1) is zero. The total density of the systems we consider is invariant over time.

Therefore, to compute the density at an arbitrary point $x_T \in \mathcal{X}$ at time T, one can simply proceed as follows: First, find the initial state $x_0 = \Phi(x_T, -T)$ using the inverse dynamics $-f$. Then, solve the Eq. (2) with initial condition $[x_0, \rho_0(x_0)]$. The solution at time T just gives the desired density value. However, such a procedure only gives the density at a single point, therefore, cannot be used in reachability analysis. Instead, we need to compute the solution of Eq. (2) for a set of initial conditions and use that to compute the reachable sets. To achieve this, we use an NN with ReLU (Rectified Linear Unit) activation functions [61] to jointly approximate the flow map Φ and the *density concentration function*, as will be shown in the next section.

3 Density learning and online reachability density computation

Let us take a closer look at Eq. (2). For an initial condition $[x_0, \rho_0(x_0)]$, the closed form of the solution ρ is $\rho(\Phi(x_0, t), t) = \rho_0(x_0) \exp\left(-\int_0^t \nabla \cdot f(\Phi(x_0, \tau))d\tau\right)$. Interestingly, the solution of ρ is linear in the initial condition ρ_0. The gained part denoted by $G(x_0, t) := \exp\left(-\int_0^t \nabla \cdot f(\Phi(x_0, \tau))d\tau\right)$ is a function of the initial state x_0 and time t, but is independent of ρ_0. This mapping G is completely determined by the underlying dynamics and we call it the *density concentration function*. With the density concentration function G in hand, the density at an arbitrary time and state can be quickly computed from any ρ_0. However, G is obviously hard to compute. Therefore, we propose to use NN to approximate the density concentration function. Moreover, in addition to G, we also use NN to approximate the flow map Φ, which will be shown in Sec. 3.2 a necessity for computing the reachability density distribution.

3.1 System dynamics and density learning framework

Let the parameterized versions of the flow map and the density concentration function be Φ_ω and G_θ respectively, where ω and θ are parameters. To train the neural network, we construct a dataset by randomly sampling N trajectories of the system: $\mathcal{D}_{train} = \{\xi_i\}_{i=0}^{N-1}$ in T time steps (with time interval Δt): $\xi_i = \{(x_0^i, 0), (x_1^i, \Delta t), ..., (x_{T-1}^i, (T-1)\Delta t)\}$ where $x_j^i = \Phi(x_0^i, j\Delta t)$. Then, the goal of the learning is to find parameters ω and θ satisfying

$$\begin{cases} \Phi_\omega(x_0^i, k\Delta t) - x_k^i = 0, \forall i, k, \\ \frac{\partial G_\theta(x_0^i, k\Delta t)}{\partial t} + G_\theta(x_0^i, k\Delta t) \cdot (\nabla \cdot f(x_k^i)) = 0, \forall i, k, \end{cases} \tag{3}$$

where the first constraint is for the flow map estimation, and the second constraint enforces the Liouville equation for all the data points.

As for the implementation, we model Φ_ω and G_θ jointly as a fully-connected neural network $\text{NN}(\cdot, \cdot)$ with ReLU activations. To ensure numerical stability as G is an exponential function, we add a nonlinear transform from the NN output to the density gain:

$$\begin{cases} G_\theta(x_0, t) = \exp(t \cdot \text{NN}_{[0:1]}(x_0, t)) = \exp(t \cdot z(x_0, t)), \\ \Phi_\omega(x_0, t) = \text{NN}_{[1:n+1]}(x_0, t), \end{cases} \tag{4}$$

where $\text{NN}_{[i:j+1]}$ is to choose the $i, i+1, ..., j$-th dimensions from the output of the NN, and z is the intermediate density estimation from the NN. In this way, we guarantee that the density gain at $t = 0$ is always 1. We optimize our NN via back propagation, using the loss function defined below:

$$\mathcal{L} = \lambda \cdot \sum_{i,k} \left[\Phi_\omega(x_0^i, k\Delta t) - x_k^i\right]^2 + \sum_{i,k} \left[\dot{G}_\theta(x_0^i, k\Delta t) + G_\theta(x_0^i, k\Delta t)\left(\nabla \cdot f(x_k^i)\right)\right]^2, \tag{5}$$

where the first term denotes the state estimator square error [62], the second term indicates how far (in the sense of L2-norm) the solution deviates from the Liouville Equation, and λ balances these two loss terms. We approximate the time derivative of the density concentration function by $\dot{G}_\theta(x_0^i, k\Delta t) = \left[G_\theta(x_0^i, (k+1)\Delta t) - G_\theta(x_0^i, k\Delta t)\right]/\Delta t$. Our method can also work for black-box systems if we approximate $\nabla \cdot f(\cdot)$ numerically from data.

With some tools from statistical learning theory, we can show that with a large enough number N of samples, the learned flow map and the density concentration function can be arbitrarily accurate. A formal proof can be found in the supplementary materials.

3.2 System reachable set distribution computation via NN Reachability Analaysis

In the last section, we have learned an NN that can estimate the density at a single point given the initial density. In this section, we will boost this single-point estimation to set-based estimation by analyzing the reachability of the learned NN. To compute the set of all reachable states and the corresponding density from a set of initial conditions, we use the Reachable Polyhedron Marching (RPM) method [21] to further process the learned NN for G and Φ. RPM is a polyhedron-based approach for exact reachability analysis for ReLU NNs. It partitions the input space into polyhedral cells such that in each cell the ReLU activation map does not change and the NN becomes a fixed affine mapping. With these polyhedral cells and the corresponding affine mapping on each cell, the reachable set for a set of input can be quickly and exactly evaluated. Backward reachability analysis can also be performed by computing the intersection of the pre-image of a query output polyhedron with the input polyhedron cells.

System forward reachable set with density. Recall that the input of the NN in Sec. 3.1 consists of the initial state x_0 and time t. For the simplicity of comparison with other methods, we only estimate the reachable set and the density at given fixed time instances t and fix the last element of the input of the NN. Thus for each time step t, the input polyhedral cells generated from the RPM will be a set of linear inequality constraints on x, and those input cells together with the set of the affine mappings and output polyhedral cells can be represented as: $\{(A_k, b_k, C_k, d_k, E_k, f_k)\}_{k=1}^N$, where in each input cell $H_k := \{v \in \mathbb{R}^d | A_k v \leq b_k\}$, the NN becomes an affine mapping $y = C_k v + d_k$, and thus the image of the input cell is also a polyhedron $M_k = \{y \in \mathbb{R}^{d+1} | E_k y \leq f_k\}$.

Recall that the first dimension of our NN output estimates the density concentration function z, and the rest dimensions estimate the state x, thus the output cell can be written as $M_k = \{(z, x) | E_k [z,\, x]^T \leq f_k\}$. By projecting it to the state space, we get the reachable set of the system, i.e., $R_k^o := \{x \in \mathcal{X} | (z, x) \in M_k\}$. Then, in each cell M_k, we evaluate the lower and upper bounds of z, and denote them by $z_{k,min}$ and $z_{k,max}$ respectively. The density bound for cell M_k is then computed as

$$\begin{cases} \rho_{k,min} = \rho_0(\bar{x}_k) \cdot e^{t \cdot z_{k,min}}, \\ \rho_{k,max} = \rho_0(\bar{x}_k) \cdot e^{t \cdot z_{k,max}}, \end{cases} \tag{6}$$

where $\bar{x}_k$ is the center of H_k.

Finally the system forward reachable set can be represented as a union of projected output polyhedral cells: $\bigcup_{k=1}^{N} R_k^o$ where each cell R_k^o is associated with a density bound $[\rho_{k,min}, \rho_{k,max}]$.

System reachable set probability computation. Given an initial state distribution, we want to figure out the probability distribution of the system forward reachable sets, as well as the probability for the states land into a query set (e.g., the query set could be the unsafe region).

For an arbitrary initial probability density function (whose support is bounded), we can apply RPM to partition its support into cells[4] as in the last section. Finally, we will obtain the reachable sets with bounded state densities $\{(H_k, \rho_k^{min}, \rho_k^{max})\}_{k=1}^N$. The probability bound in each cell can be further computed as

$$\begin{cases} P_k^{min} = \text{Vol}(H_k)\rho_k^{min}, \\ P_k^{max} = \text{Vol}(H_k)\rho_k^{max}, \end{cases} \tag{7}$$

where $\text{Vol}(\cdot)$ computes the volume for a polyhedron. By computing for all input cells $\{H_k\}_{k=1}^N$, we can derive the system forward reachable set and the corresponding probability bound as $\left\{(H_k, P_k^{min}, P_k^{max})\right\}_{k=1:N}$. The backward reachable set probability can be computed in a similar fashion, by checking the intersection between the query output region and the output cells derived by RPM, computing each intersection's probability range by its volume and the density bound, and finally aggregating the probability of all those intersections. The detailed computation is provided in the supplementary material.

[4] We can always further divide those input cells to make the bound tighter/ more precise, while still guaranteeing the Neural Network on each cell can be seen as an affine transformation.

Table 1: KL divergence comparing to ODE (2)-simulated density distribution.

System	Dim.	Control	KDE	Histogram	Ours	Error reduction (Ours vs. KDE)
Van der Pol Oscillator	2	-	3.9630	3.7315	0.3248	91.80%
Double integrator [39]	2	NN	0.0940	0.1278	0.0656	30.13%
Kraichnan-Orszag system [64]	3	-	0.2672	0.2910	0.0009	99.67%
Inverted pendulum [65]	4	LQR	6.0624	5.2921	0.3609	94.05%
Ground robot navigation	4	NN	1.1188	1.5139	0.1560	86.05%
FACTEST car tracking system [66]	5	Tracking	0.2324	0.5703	0.0082	96.45%
Quadrotor control system [39]	6	NN	0.0519	2.0507	0.0208	59.89%
Adaptive cruise control system [4]	7	NN	2.7616	N/A*	0.0143	99.48%
F-16 Ground collision avoidance [67]	13	Hybrid	8.2560	N/A*	0.0104	99.87%
8-Car platoon system [68]	16	NN	1.1084	N/A*	0.5466	50.69%

4 Experimental evaluation in simulation

In this section, we conduct experiments to study the advantages of learning state density concentration function from Liouville PDE in density estimation, and in the reachable set distribution analysis. More experiments and details will be provided in the supplementary material.

4.1 Implementation details

Datasets: We investigate 10 different benchmark dynamical systems, as reported in the first two columns of Table 1. These benchmark systems range from low dimension academic models (2$\sim$3 dimensions) to complex and even black-box systems (13$\sim$18 dimensions) controlled by handcrafted or NN controllers. All controllers (except for the ground robot navigation example) are from the original references. More details, including the model description and initial distributions, will be provided in the supplementary material.

Training: For each system, we generate 10k trajectories through simulation with varied trajectory lengths from 10 to 100 time-steps, depending on different configurations of the simulation environment. We use 80% of the samples to train the NN as described in Sec. 3.1 and use the rest for validation. For the trajectory data, we collect the system states and compute for the system divergence term. For black-box systems, we use the gradient perturbation method to approximate the derivatives. We use feed-forward NN which has 3 hidden layers with 64 hidden units in each layer. We use PyTorch [63] to train the NN and the training takes around 1$\sim$2 hours on an RTX2080 Ti GPU.

4.2 Density estimation verification

We first test the density estimation accuracy of our learned NN. We compare our approach with other density estimation approaches including kernel density estimation (KDE) and the histogram approach. For each simulation scenario, we first solve Eq. (2) to generate 20k $\sim$ 100k trajectories of (state, density) pairs, and treat this density value as the ground truth. For the KDE method, we choose an Epanechnikov kernel. We then measure the KL divergence between the density estimate of each method and the ground truth. As shown in Table. 1, our approach has consistently outperformed KDE and histogram-based approaches, with the largest reduction of 99.69% in KL divergence when compared with the histogram approach for the Kraichnan-Orszag system, while our approach doesn't use any ODE generated density values during training. Also in high-dimension systems (state dimension $\geq$ 7), the histogram approach fails to predict the density due to the curse of the dimensionality, whereas our approach can always predict the density, with a 30.13% to 99.87% decrease in KL divergence comparing to KDE. More visualizations for the plots will be given in the supplementary materials.

4.3 Reachable set distribution analysis

Forward reachable set distribution analysis Being confident that our approach is able to provide an accurate state density estimation, we extend our NN to do distribution analysis, which is a valuable technique in safety-related applications like autonomous driving. Here we use an existing

reachability tool RPM [21] to compute the forward reachable sets with probability bounds. Details about how to derive the density and probability bound for the reachable sets are presented in Sec. 3.2 and in the supplemental materials. Also, RPM was only able to parse the NN for Van der Pol, Double integrator, ground robot navigation, and the FACTEST car model. It fails in handling other high-dimension complex systems due to numerical issues when partitioning for the input polyhedral cells. Therefore, we only report the results on those 4 models in this section. The main purpose of this section is to show that for some systems the density tends to concentrate on certain states. In this case, a small portion of the reachable sets contains the major set of states that are more likely to be reached. Therefore, our method can better quantify risks than worst-case reachability, by providing a flexible threshold for the probability of reachability.

We start with the Van der Pol oscillator. The initial states are uniformly sampled from a square region: $[-2.5, 2.5] \times [-2.5, 2.5]$. As illustrated in Fig. 1, the system states will gradually converge to a limit cycle. Using worst-case reachability analysis for this system will result in a very conservative over-approximation, and this over-approximation will propagate over time and lead to increasing inaccuracy of the reachable set estimation. As shown in Fig. 2(a), for the worst-case methods like DryVR(red) [66] and GSG(Green) [69], the volume of their estimated reachable set relative to the volume of the convex hull of the system states keeps increasing over time, from 1.9670X to 5.3027X for the DryVR [66] approach, and from 1.7636X to 2.8086X for GSG [69]. However, our method can give the probability bound for every reachable set in the state space as shown in the heatmap in Fig. 2(b), clearly identifying the region around the limit cycle in high density (bright color), and the rest space in low density (dark color).

We can also compute the relative volume of the reachable sets (comparing to the convex hull) preserving different levels of reachable probability, whose evolution over time reflects the system's tendency for concentration. As shown in Fig. 3, we use the above 4 systems and study the volume of the reachable set with probability threshold 0.50, 0.70, 0.80, 0.90, and 0.99. As expected, the relative volume will increase as the probability threshold increases. In all cases, there exist some time instances where a small volume of the reachable set actually preserves high probability, which shows the state concentration exists in many existing systems. While as shown in Fig. 3, the worst-case reachability tools can only generate one curve which presents the (relative) volume of the reachable set that covers all possible states. Not surprisingly, the worst-case reachability tools give very conservative results. We believe using our proposed method to do reachable set distribution analysis will benefit future study for systems with uncertainty, and for systems where the failure case is inevitable but happens with a low probability.

Online safety verification under different initial state distributions Since our approach learns the density concentration function instead of absolute value for the density, it has the flexibility to perform online reachable set distribution analysis with any possible (bounded support) initial state distributions. Here we consider the safety verification problem of the ground robot navigation problem, where the obstacle is in the center of the environment. The obstacle can be represented by a polyhedron, and the probability of reaching this obstacle is determined by the initial state distribution (see how to compute it in Sec. 3.2). We parametrized the initial state distribution as a (truncated) Gaussian distribution $N(\mu, \sigma^2 I)$, where μ is the initial state of the robot, and σ measures the uncertainty of the initial state. We use our approach to compute the upper bound and lower bound for the probability that the robot will reach the obstacle. For 50 time steps, our reachability distribution analysis process can be done in less than 20ms per step with parallel computation and heuristic searching (see details in the supplementary material). As shown in Fig. 4, when σ decreases from 1.0 to 0.02, the upper and lower bounds for the probability of reaching the unsafe obstacle decrease to close to zero (from 0.04236 to 3.5344×10^{-06}), where the worst-case reachability analysis will only report a collision is inevitable, without quantifying the corresponding risks. This also clearly verifies that the learned

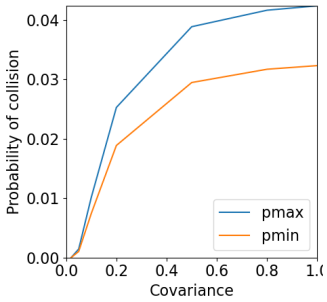
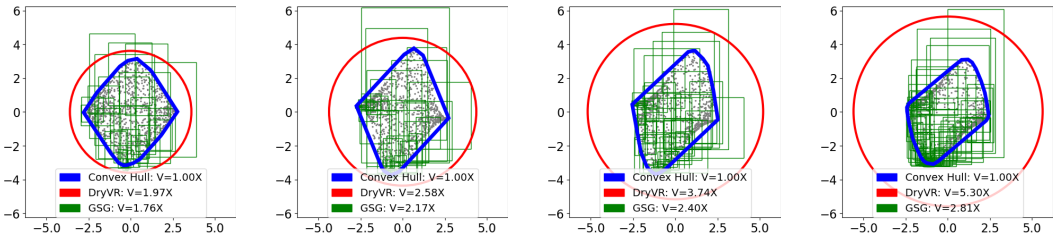
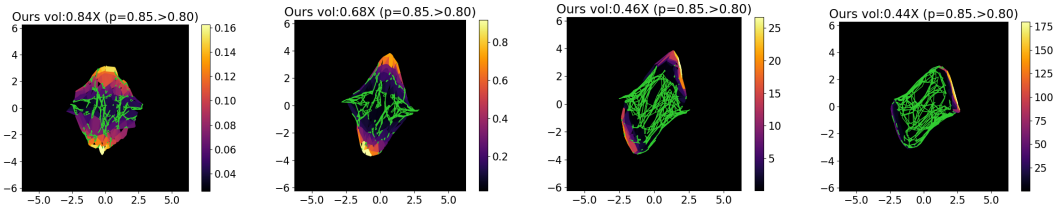


Figure 4: Probabilitstic safety verification under different initial state distributions. As the vehicle has less uncertainty (covariance) in its initial position and velocity, it will have lower probability to collide.

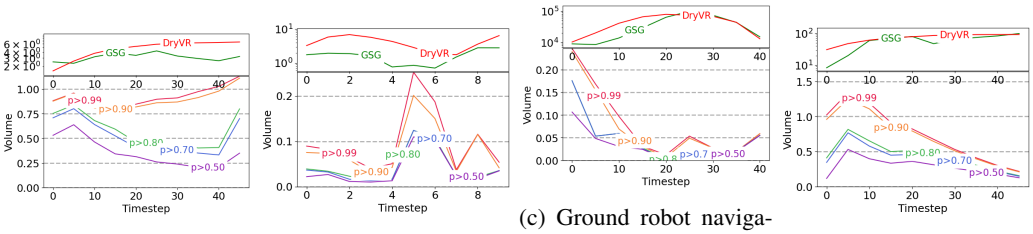


(a) The system forward reach set computed by the ConvexHull method(blue) [9], DryVR(red) [66] and GSC(green) [69]. We further show the relative volume of each kind of reachable set, comparing to the volume of the convex hull of the sampled points (gray). As time evolves, the conventional reachability methods often result in a larger over-approximation of the reachable sets with an increasing estimation error.



(b) The system forward reach set derived by RPM. The reachable sets are represented in polyhedral cells. The color ranged from dark purple to light yellow indicates the density inside the polyhedral cells. The edges colored in green indicate the boundaries of the RPM polyhedral cells with density below a threshold.

Figure 2: The Van der Pol oscillator forward reachable set comparison between the worst-case reachability methods [66, 69] in Fig. 2(a) and our probabilistic approach in Fig. 2(b). Our approach clearly identifies the reachable sets in high and low densities, and shows that as time evolves, the states will concentrate on the limit cycle, which is as expected.



(a) Van der Pol system (b) Double integrator (c) Ground robot navigation (d) FACTEST car model

Figure 3: The (relative) volume of reachable set with different probability level using our method, and comparison to other methods. The top part of each figure is in logarithm scale.

density concentration function is independent of the initial distribution. With the learned density concentration function, reachability analysis can be done quickly for an arbitrary initial distribution.

5 Conclusion and discussion

In this paper, we propose a Neural Network-based probabilistic safety verification framework that can compute reachable sets as well as their corresponding probability distribution. Our approach is data-driven, can handle nonlinear (and potentially black-box) systems with varying initial state distributions, and can be used for fast online safety verification. We believe that method is more helpful than worst-case reachability particularly when the system states are more likely to concentrate. One limitation of our approach is that the NN reachability tool we used (RPM) cannot handle high-dimension complex systems or cases where the support of the initial distribution is very large, due to the numerical issues when partitioning for polyhedral cells. We believe this can be solved by doing input normalization or using other reachability tools. This will be left as our future work. We also plan to explore more complicated systems (e.g. high-dimension systems, hybrid systems) in real-world applications.

References

[1] M. Chen and C. J. Tomlin. Hamilton–jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:333–358, 2018.

[2] A. Devonport and M. Arcak. Data-driven reachable set computation using adaptive gaussian process classification and monte carlo methods. In *2020 American Control Conference (ACC)*, pages 2629–2634. IEEE, 2020.

[3] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[4] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, pages 3–17. Springer, 2020.

[5] S. Bansal and C. Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. *arXiv preprint arXiv:2011.02082*, 2020.

[6] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*, 2019.

[7] A. Devonport and M. Arcak. Estimating reachable sets with scenario optimization. In *Learning for Dynamics and Control*, pages 75–84. PMLR, 2020.

[8] L. Liebenwein, C. Baykal, I. Gilitschenski, S. Karaman, and D. Rus. Sampling-based approximation algorithms for reachability analysis with provable guarantees. *RSS*, 2018.

[9] T. Lew and M. Pavone. Sampling-based reachability analysis: A random set theory approach with adversarial sampling. *arXiv preprint arXiv:2008.10180*, 2020.

[10] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 202–214. Springer, 2000.

[11] A. Girard. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 291–305. Springer, 2005.

[12] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. In *International Conference on Computer Aided Verification*, pages 477–494. Springer, 2016.

[13] P.-J. Meyer, A. Devonport, and M. Arcak. Tira: Toolbox for interval reachability analysis. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 224–229, 2019.

[14] S. Bak. Reducing the wrapping effect in flowpipe construction using pseudo-invariants. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, pages 40–43, 2014.

[15] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *2008 47th IEEE Conference on Decision and Control*, pages 4042–4048. IEEE, 2008.

[16] J. Cyranka, M. A. Islam, G. Byrne, P. Jones, S. A. Smolka, and R. Grosu. Lagrangian reachabililty. In *International Conference on Computer Aided Verification*, pages 379–400. Springer, 2017.

[17] M. Ehrendorfer. The liouville equation and prediction of forecast skill. In *Predictability and Nonlinear Modelling in Natural Sciences and Economics*, pages 29–44. Springer, 1994.

[18] T. Nakamura-Zimmerer, D. Venturi, Q. Gong, and W. Kang. Density propagation with characteristics-based deep learning. *arXiv preprint arXiv:1911.09311*, 2019.

[19] Y. Chen, M. Ahmadi, and A. D. Ames. Optimal safe controller synthesis: A density function approach. In *2020 American Control Conference (ACC)*, pages 5407–5412. IEEE, 2020.

[20] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[21] J. A. Vincent and M. Schwager. Reachable polyhedral marching (rpm): A safety verification algorithm for robotic systems with deep neural network components. *arXiv preprint arXiv:2011.11609*, 2020.

[22] International competition on verifying continuous and hybrid systems. `https://cps-vo.org/group/ARCH/FriendlyCompetition`. Accessed: 2021-06-18.

[23] G. Agha and K. Palmskog. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 28(1):1–39, 2018.

[24] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50 (7):947–957, 2005.

[25] B. Xue, M. Zhang, A. Easwaran, and Q. Li. PAC model checking of black-box continuous-time dynamical systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3944–3955, 2020.

[26] A. Berndt, A. Alanwar, K. H. Johansson, and H. Sandberg. Data-driven set-based estimation using matrix zonotopes with set containment guarantees. *arXiv preprint arXiv:2101.10784*, 2021.

[27] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone. A machine learning approach for real-time reachability analysis. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 2202–2208. IEEE, 2014.

[28] M. Rasmussen, J. Rieger, and K. N. Webster. Approximation of reachable sets using optimal control and support vector machines. *Journal of Computational and Applied Mathematics*, 311:68–83, 2017.

[29] A. J. Thorpe, K. R. Ortiz, and M. M. Oishi. Data-driven stochastic reachability using hilbert space embeddings. *arXiv preprint arXiv:2010.08036*, 2020.

[30] A. Chakrabarty, C. Danielson, S. Di Cairano, and A. Raghunathan. Active learning for estimating reachable sets for systems with unknown dynamics. *IEEE Transactions on Cybernetics*, 2020.

[31] A. R. R. Matavalam, U. Vaidya, and V. Ajjarapu. Data-driven approach for uncertainty propagation and reachability analysis in dynamical systems. In *2020 American Control Conference (ACC)*, pages 3393–3398. IEEE, 2020.

[32] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[33] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pages 443–452. Springer, 2019.

[34] W. Xiang, H.-D. Tran, and T. T. Johnson. Output reachable set estimation and verification for multilayer neural networks. *IEEE transactions on neural networks and learning systems*, 29 (11):5777–5783, 2018.

[35] X. Yang, H.-D. Tran, W. Xiang, and T. Johnson. Reachability analysis for feed-forward neural networks using face lattices. *arXiv preprint arXiv:2003.01226*, 2020.

[36] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

[37] J. Fan, C. Huang, X. Chen, W. Li, and Q. Zhu. Reachnn*: A tool for reachability analysis of neural-network controlled systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 537–542. Springer, 2020.

[38] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5929–5934. IEEE, 2020.

[39] M. Everett, G. Habibi, and J. P. How. Efficient reachability analysis of closed-loop systems with neural network controllers. *arXiv preprint arXiv:2101.01815*, 2021.

[40] D. A. Spencer and R. D. Braun. Mars pathfinder atmospheric entry-trajectory design and dispersion analysis. *Journal of Spacecraft and Rockets*, 33(5):670–676, 1996.

[41] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.

[42] C. Pantano and B. Shotorban. Least-squares dynamic approximation method for evolution of uncertainty in initial conditions of dynamical systems. *Physical Review E*, 76(6):066705, 2007.

[43] H. Lee and I. S. Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

[44] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[45] T. Uchiyama and N. Sonehara. Solving inverse problems in nonlinear pdes by recurrent neural networks. In *IEEE International Conference on Neural Networks*, pages 99–102. IEEE, 1993.

[46] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[47] J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[48] E. Weinan and B. Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[49] J. He, L. Li, J. Xu, and C. Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

[50] J. Han, A. Jentzen, et al. Algorithms for solving high dimensional pdes: From nonlinear monte carlo to machine learning. *arXiv preprint arXiv:2008.13333*, 2020.

[51] Y. Khoo, J. Lu, and L. Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.

[52] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, pages 1–53, 2021.

[53] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[54] L. Yang, D. Zhang, and G. E. Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *arXiv preprint arXiv:1811.02033*, 2018.

[55] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[56] Z. Long, Y. Lu, X. Ma, and B. Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216. PMLR, 2018.

[57] L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[58] A. Koryagin, R. Khudorozkov, and S. Tsimfer. Pydens: A python framework for solving differential equations with neural networks. *arXiv preprint arXiv:1909.11544*, 2019.

[59] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[60] F. Chen, D. Sondak, P. Protopapas, M. Mattheakis, S. Liu, D. Agarwal, and M. Di Giovanni. Neurodiffeq: A python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46):1931, 2020.

[61] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[62] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3): 1247–1250, 2014.

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[64] S. A. Orszag and L. Bissonnette. Dynamical properties of truncated wiener-hermite expansions. *The Physics of Fluids*, 10(12):2603–2613, 1967.

[65] Y.-C. Chang, N. Roohi, and S. Gao. Neural lyapunov control. *arXiv preprint arXiv:2005.00611*, 2020.

[66] C. Fan, K. Miller, and S. Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. In *International Conference on Computer Aided Verification*, pages 629–652. Springer, 2020.

[67] P. Heidlauf, A. Collins, M. Bolender, and S. Bak. Verification challenges in f-16 ground collision avoidance and other automated maneuvers. In *ARCH@ ADHS*, pages 208–217, 2018.

[68] H. Zhu, Z. Xiong, S. Magill, and S. Jagannathan. An inductive synthesis framework for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 686–701, 2019.

[69] M. Everett, G. Habibi, and J. P. How. Robustness analysis of neural networks via efficient partitioning with applications in control systems. *IEEE Control Systems Letters*, 2020.