GoTube: Scalable Statistical Verification of Continuous-Depth Models

**Sophie Gruenbacher [1*] , Mathias Lechner [2], Ramin Hasani [3],
Daniela Rus [3], Thomas A. Henzinger [2], Scott A. Smolka [4], Radu Grosu [1]**

Abstract

We introduce a new statistical verification algorithm that formally quantifies the behavioral robustness of any time-continuous process formulated as a continuous-depth model. Our algorithm solves a set of global optimization (Go) problems over a given time horizon to construct a tight enclosure (Tube) of the set of all process executions starting from a ball of initial states. We call our algorithm GoTube. Through its construction, GoTube ensures that the bounding tube is conservative up to a desired probability and up to a desired tightness. GoTube is implemented in JAX and optimized to scale to complex continuous-depth neural network models. Compared to advanced reachability analysis tools for time-continuous neural networks, GoTube does not accumulate overapproximation errors between time steps and avoids the infamous wrapping effect inherent in symbolic techniques. We show that GoTube substantially outperforms state-of-the-art verification tools in terms of the size of the initial ball, speed, time-horizon, task completion, and scalability on a large set of experiments. GoTube is stable and sets the state-of-the-art in terms of its ability to scale to time horizons well beyond what has been previously possible.

Introduction

The use of deep-learning systems powered by continuous-depth models continues to grow, especially due to the revival of neural ordinary differential equations (Neural ODEs) (Chen et al. 2018). These models parametrize the derivative of the hidden states by a neural network. The resulting system of differential equations can perform strong function approximation and generative modeling. Ensuring their safety and robustness in any of these fronts is a major imperative, particularly in high-stakes decision-making applications such as medicine, automation, and finance.

A particularly appealing approach is to construct a tight overapproximation of the set of states reached over time according to the neural network's dynamics (a bounding tube) and provide deterministic or stochastic guarantees for the conservativeness of the tube's bounds.

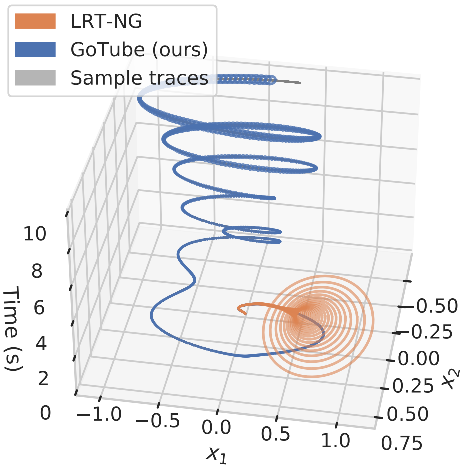

Figure 1: Reachtubes of LRT-NG (Gruenbacher et al. 2020) and GoTube for a CT-RNN controlling CartPole-v1 environment. CAPD (Kapela et al. 2020) and Flow* (Chen, Ábrahám, and Sankaranarayanan 2013) failed.

Since all these networks represent nonlinear systems of ordinary differential equations, it is impossible, that is, undecidable, to exactly predict their behavior, as they do not have a closed-form solution. One is able to calculate the solution for different initial states, but one does not know what happens in between of these already calculated traces. The main goal in the reachability analysis of nonlinear ODEs, is to overapproximate the reachable states of the ODEs, starting from a set of initial states, such as, an interval, a ball, or an ellipsoid, in a way that one can guarantee that all traces are inside the overapproximation. We call such an overapproximation a *bounding tube*.

Deterministic verification approaches ensure conservative bounds (Chen, Ábrahám, and Sankaranarayanan 2013; Gowal et al. 2018; Mirman, Gehr, and Vechev 2018; Bunel et al. 2020a; Kapela et al. 2020; Gruenbacher et al. 2020), but often sacrifice speed and accuracy (Ehlers 2017), and thus scalability; see CAPD, Flow*, and LRT-NG in Fig. 1 and Fig. 3. Statistical methods, on the other hand, only ensure a weaker notion of conservativeness in the form of confidence intervals (statistical bounds). This, however, al-

[*1] TU Wien, [2] IST Austria, [3] CSAIL MIT, [4] Stony Brook University.
Correspondence to: sophie.gruenbacher@tuwien.ac.at
Code: https://github.com/DatenVorsprung/GoTube

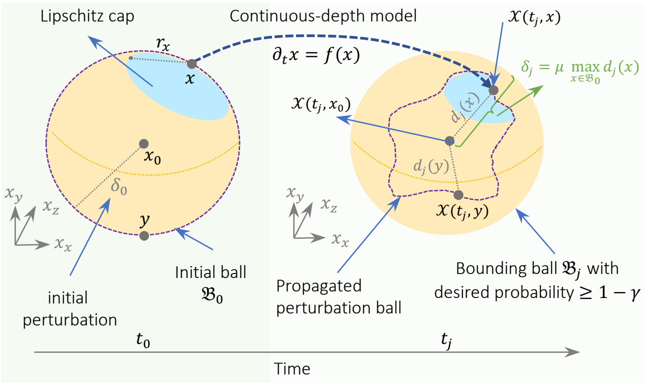

Figure 2: GoTube in a nutshell. The center x_0 of ball $\mathcal{B}_0 = B(x_0, \delta_0)$, with δ_0 the initial perturbation, and samples x drawn uniformly from $\mathcal{B}_0$'s surface, are numerically integrated in time to $\chi(t_j, x_0)$ and $\chi(t_j, x)$, respectively. The Lipschitz constant of $\chi(t_j, x)$ and their distance $d_j(x)$ to $\chi(t_j, x_0)$ are then used to compute Lipschitz caps around samples x, and the radius δ_j of bounding ball $\mathcal{B}_j$ depending on the chosen tightness factor μ. The ratio between the caps' surfaces and $\mathcal{B}_0$'s surface are correlated to the desired confidence $1 - \gamma$.

lows them to achieve much more accurate and faster verification algorithms that scale up to much larger dynamical system (Shmarov and Zuliani 2015b; Bortolussi and Sanguinetti 2014; Gruenbacher et al. 2021).

It was recently shown theoretically that statistical verification approaches based on Lagrangian reachability (SLR) could provably guarantee confidence intervals for continuous-depth models (Gruenbacher et al. 2021). The proposed theoretical framework suggests performing both statistical global optimization and local differential optimization (Zhigljavsky and Zilinskas 2008; Pontryagin 2018), and uses *interval arithmetic* to symbolically bound the Lipschitz constant. Thus, it can construct a bounding ball of the reachable states at every time step, and over time, a tight bounding Tube. Although these theoretical results suggest an elegant way to avoid compounding errors, the SLR algorithm has not been implemented, so is this approach computationally tractable in practice?

We implemented the SLR algorithm as instructed in (Gruenbacher et al. 2021). We observed that even after resolving the first-occurring inefficient sampling and their vanishing gradient problems, the algorithm still blew up in time, even for low-dimensional benchmarks such as the Dubins Car. There are three fundamental algorithmic constraints of the symbolic techniques such as stochastic Lagrangian reachability that result in them being computationally intractable: 1) the use of interval arithmetic for computing a conservative upper bound for the Lipschitz constant of the system fundamentally limits the scalability of reachability-based verification methods, 2) the use of local gradient descent to search for local maxima in practice is more expensive than a simple local search, and 3) the computational overhead due to the propagation of many initial states is high.

In this work, we propose technical solutions for these fundamental issues and introduce a practical statistical verification algorithm for continuous-time models. In particular, to tackle the first fundamental challenge introduced above, we develop a new theory that allows us to compute statistical bounds for the Lipschitz constant in order to define a spherical cap around each sample, where the maximum perturbation is stochastically bounded (Lipschitz cap). As such, we are able to remove the conservative interval-based computation of the Lipschitz constant. Furthermore, we provide convergence guarantees for computing the upper bound of the confidence interval for the maximum perturbation at time t_j with confidence level $1 - \gamma$ and tube tightness μ, using the estimation of the Lipschitz constant. This eliminates the dependence on the propagation horizon and considerably reduces computational complexity in the number of samples. We directly use this new Lipschitz constant computation framework instead of the costly interval arithmetic.

We supply our global optimization scheme with a simple sampling process to propagate the initial states in parallel, according to the neural network's dynamics. This compensates for local differential optimization with additional samples. Our algorithm is called GoTube, as it solves a set of global optimization problems to construct a tight and computationally tractable enclosure (Tube) of all possible evolutions of the system for a given time horizon.

GoTube takes advantage of advanced automatic differential toolboxes such as JAX to perform highly parallel and tensorized operations to further enhance the runtime of the verification suite. On a large set of experiments with continuous-depth models, GoTube substantially outperforms state-of-the-art verification tools in terms of the size of the initial ball, speed, time-horizon, task completion, and scalability. We summarize the contributions of our paper as follows:

- A novel and efficient theory for computing statistical bounds for the Lipschitz constant of the system, which helps us achieve tight reachtubes for continuous-time dynamical systems.
- We prove convergence guarantees for the GoTube Algorithm, thus ensuring that the algorithm terminates in finite time even using stochastic Lipschitz caps around the samples instead of deterministic local balls.
- We perform a diverse set of experiments on continuous-time models with increasing complexity and demonstrate that GoTube considerably outperforms state-of-the-art verification tools.

Related Work

Global Optimization. Efficient local optimization methods such as gradient descent cannot be used for global optimization since such problems are typically non-convex. Thus, many advanced verification algorithms tend to use global optimization schemes (Bunel et al. 2018, 2020a). Depending on the properties of the objective function, e.g. smoothness, various types of global optimization techniques exist. For instance, interval-based branch-and-bound (BaB) algorithms (Neumaier 2004; Hansen and Walster 2003) work well on differentiable objectives up to a certain scale, which has recently been improved (De Palma et al. 2021). There are also

Table 1: Related work on the reachability analysis of continuous-time systems. Determ.= Deterministic. ”No” indicates a stochastic method. Table content is partially reproduced from (Gruenbacher et al. 2021).

Technique	**Determ.**	**Parallel**	**wrapping effect**	**Arbitrary Time-horizon**
LRT (Cyranka et al. 2017) with Infinitesimal strain theory	yes	no	yes	no
CAPD (Kapela et al. 2020) implements Lohner algorithm	yes	no	yes	no
Flow-star (Chen, Ábrahám, and Sankaranarayanan 2013) with Taylor models	yes	no	yes	no
δ-reachability (Gao, Kong, and Clarke 2013) with approximate satisfiability	yes	no	yes	no
C2E2 (Duggirala et al. 2015) with discrepancy functions	yes	no	yes	no
LDFM (Fan et al. 2017) by simulation, matrix measures	yes	yes	no	no
TIRA (Meyer, Devonport, and Arcak 2019) with second-order sensitivity	yes	yes	no	no
Isabelle/HOL (Immler 2015) with proof-assistant	yes	no	yes	no
Breach (Donzé 2010; Donzé and Maler 2007) by simulation	yes	yes	no	no
PIRK (Devonport et al. 2020) with contraction bounds	yes	yes	no	no
HR (Li, Bak, and Bogomolov 2020) with hybridization	yes	no	yes	no
ProbReach (Shmarov and Zuliani 2015a) with δ-reachability,	no	no	yes	no
VSPODE (Enszer and Stadtherr 2011) using p-boxes	no	no	yes	no
Gaussian process (GP) (Bortolussi and Sanguinetti 2014)	no	no	no	no
Stochastic Lagrangian reachability SLR (Gruenbacher et al. 2021)	no	yes	no	no
GoTube (Ours)	no	yes	no	**yes**

Lipschitz-global optimization methods for satisfying Lipschitz conditions (Malherbe and Vayatis 2017; Kvasov and Sergeyev 2013). For example, a method for computing the Lipschitz constant of deep neural networks to assist with their robustness and verification analyses was recently proposed in (Fazlyab et al. 2019) and (Bhowmick, D'Souza, and Raghavan 2021). Additionally, there are evolutionary strategies for global optimization using the covariance matrix computation (Hansen and Ostermeier 2001; Igel, Hansen, and Roth 2007). In our approach, for global optimization, we use random sampling and compute neighborhoods (Lipschitz caps) of the samples, where we have probabilistic knowledge about the values, such that we are able to correspondingly estimate the stochastic global optimum with high confidence. (Zhigljavsky and Zilinskas 2008).

Verification of Neural Networks. A large body of work tried to enhance the robustness of neural networks against adversarial examples (Goodfellow, Shlens, and Szegedy 2014). There are efforts that show how to break the many defense mechanisms proposed (Athalye, Carlini, and Wagner 2018; Lechner et al. 2021), until the arrival of methods for formally verifying robustness to adversarial attacks around neighborhoods of data (Henzinger, Lechner, and Zikelic 2021). The majority of these complete verification algorithms for neural networks work on piece-wise linear structures of small-to-medium-size feedforward networks (Salman et al. 2019). For instance, (Bunel et al. 2020b) has recently introduced a BaB method that outperforms state-of-the-art verification methods (Katz et al. 2017; Tjandraatmadja et al. 2020). A more scalable approach for rectified linear unit (ReLU) networks (Nair and Hinton 2010) was recently proposed based on Lagrangian decomposition; this approach significantly improves the speed and tightness of the bounds (De Palma et al. 2021). The proposed approach not only improves the tightness of the bounds but also performs a novel branching that matches the performance of the learning-based methods (Lu and Mudigonda 2020) and outperforms state-of-the-art methods (Zhang et al. 2018; Singh et al. 2020; Bak et al. 2020; Henriksen and Lomuscio 2020). While these verification approaches work well for feedforward networks with growing complexity, they are not suitable for recurrent and continuous neural network instances, which we address in this work.

Verification of Continuous-time Systems. Reachability analysis is a verification approach that provides safety guarantees for a given continuous dynamical system (Gurung et al. 2019; Vinod and Oishi 2021). Most dynamical systems in safety-critical applications are highly nonlinear and uncertain in nature (Lechner et al. 2020). The uncertainty can be in the system's parameters (Wang et al. 2015; Shmarov and Zuliani 2015b; Enszer and Stadtherr 2011), or their initial state (Enszer and Stadtherr 2011; Huang et al. 2017). This is often handled by considering balls of a certain radius around them. Nonlinearity might be inherent in the system dynamics or due to discrete mode-jumps (Fränzle et al. 2011). We provide a summary of methods developed for the reachability analysis of continuous-time ODEs in Table 1.

A fundamental shortcoming of the majority of the methods described in Table 1 is their lack of scalability while providing conservative bounds. In this paper, we show that GoTube establishes the state-of-the-art for the verification of ODE-based systems in terms of speed, time-horizon, task completion, and scalability on a large set of experiments.

Setup

In this section, we introduce our notation, preliminary concepts, and definitions required to state and prove the statistical bounds that GoTube guarantees for time-continuous process models.

Continuous-depth models. These are a special case of nonlinear ordinary differential equations (ODEs), where the model is defined by the derivative of the unknown states x computed by a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^n$, which is assumed to be Lipschitz-continuous and forward-complete:

$$\partial_t x = f(x), \quad x(t_0) \in \mathcal{B}_0 = B(x_0, \delta_0), \tag{1}$$

$\mathcal{B}_0$ defines the initial ball (a region of initial states, whose radius quantifies the magnitude δ_0 of a perturbation of its center x_0). Time dependence can be incorporated by an additional variable x with $\delta_t x = 1$. Thus this definition naturally extends to time-varying ODEs. Nonlinear ODEs do not have in general closed-form solutions, and therefore one can not compute symbolically the solution $\chi(t_j, x)$ for all $x \in \mathcal{B}_0$. For a sequence of k timesteps from time t_0 until time horizon T: $t_0 < \ldots < t_k = T$, we use numerical ODE solvers to compute $\chi(t_j, x)$ of the initial value problem (IVP) in Eq. (1) at time t_j starting at different points $x(t_0) = x$.

We extend this computation to the entire ball by numerically integrating the center x_0 and a set of points $x \in \mathcal{V}$, uniformly sampled from the surface of the ball, and using this information to compute statistical upper bounds for the possible evolutions of the system. We define the bounding ball and bounding tube as follows:

Definition 1 (Bounding Ball) *Given an initial ball $\mathcal{B}_0 = B(x_0, \delta_0)$, we call $\mathcal{B}_j = B(\chi(t_j, x_0), \delta_j(\mathcal{B}_0))$ a* bounding ball *at time t_j, if it statistically bounds the reachable states x at time t_j for all initial points around x_0 having the maximal initial perturbation δ_0.*

As we do not only want to bound the perturbation at one specific time, but on a time series, we define:

Definition 2 (Bounding Tube) *Given an initial ball $\mathcal{B}_0 = B(x_0, \delta_0)$ and bounding balls for $t_0 < \ldots < t_k = T$, we call the series of bounding balls $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$ a* bounding tube.

Maximum perturbation at time t_j. To compute a bounding tube, we have to compute at every timestep t_j the maximum perturbation δ_j, which is defined as a solution of the optimization problem:

$$\delta_j \geq \max_{x \in \mathcal{B}_0} \|\chi(t_j, x) - \chi(t_j, x_0)\| = \max_{x \in \mathcal{B}_0} d(t_j, x), \tag{2}$$

where $d_j(x) = d(t_j, x)$ denotes the *distance* at time t_j, if the initial center x_0 is known from the context. As stated in (Gruenbacher et al. 2021), the radius at time t_j can be overapproximated by solving a global optimization problem on the surface of the initial ball $\mathcal{B}_0$: as we require Lipschitz-continuity and forward-completeness of the ODE in Eq. (1), the map $x \mapsto \chi(t_j, x)$ is a homeomorphism and commutes with closure and interior operators. In particular, the image of the boundary of the set $\mathcal{B}_0$ is equal to the boundary of the image $\chi(t_j, \mathcal{B}_0)$. Thus, Eq. (2) has its optimum on the surface of the initial ball $\mathcal{B}_0^S = \text{surface}(\mathcal{B}_0)$, and we will only consider points on the surface.

Main Results

Our GoTube algorithm and its theory solve fundamental scalability problems of related works (see Table 1) by replacing interval arithmetic used to compute deterministic caps with stochastic Lipschitz caps. This enables us to verify continuous-depth models up to an arbitrary time-horizon, a capability beyond what was achievable before.

Algorithm 1: GoTube

Require: initial ball $\mathcal{B}_0 = B(x_0, \delta_0)$, time horizon T, sequence of timesteps t_j ($t_0 < \cdots < t_k = T$), error tolerance $\mu > 1$, confidence level $\gamma \in (0, 1)$, batch size b, distance function d

```
$\mathcal{V} \leftarrow \{\}$    (list of visited random points)
sample batch $x^B \in \mathcal{B}_0^S$
for $(j = 1; j \leq k; j = j + 1)$ do
    $\bar{p} \leftarrow 0$
    while $\bar{p} < 1 - \gamma$ do
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{x^B\}$
        $x_j \leftarrow \chi(t_j, x_0)$    (integrate initial center point)
        $\bar{m}_{j,\mathcal{V}} \leftarrow \max_{x \in \mathcal{V}} d(t_j, x)$
        compute local Lipschitz constants $\lambda_x$ for $x \in \mathcal{V}$
        compute statistical quantile $\Delta\lambda_{\mathcal{V}}$
        compute cap radii $r_x(\lambda_x, \Delta_\lambda, \bar{m}, \mu)$ (Thm. 1) for
        $x \in \mathcal{V}$
        $\mathcal{S} \leftarrow \bigcup_{x \in \mathcal{V}} B(x, r_x)^S$    (total covered area)
        $\bar{p} \leftarrow$ computeProb$(\gamma, \{r_x : x \in \mathcal{V}\}, n, \delta_0)$
        sample batch $x^B \in \mathcal{B}_0$
    end while
    $\delta_j \leftarrow \mu \cdot \bar{m}_{j,\mathcal{V}}$
    $\mathcal{B}_j \leftarrow B(x_j, \delta_j)$
end for
return $(\mathcal{B}_1, \ldots, \mathcal{B}_k)$
```

To be able to do that, we formulated Theorems on: 1) How to choose the radius of a Lipschitz cap using probabilistic bounds of local Lipschitz constants of the samples together with a statistical quantile. 2) Convergence guarantees using these new stochastic caps, as they are used by GoTube to compute the probability of δ_j being an upper bound of the biggest perturbation. In addition, we implemented tensorization and substantially increased the number of random samples, thus being able to remove the dependence on the propagation-horizon of the gradient descent and increasing the computation speed to be able to deal with continuous-depth models.

We start by describing the GoTube Algorithm. This facilitates the comprehension of the different computation and theory steps. Given a continuous-depth model as in Eq. (1), an initial ball $\mathcal{B}_0$ defined by a center point x_0 and the maximum initial perturbation δ_0, a time horizon T with a sequence of timesteps t_j ($t_0 < \ldots < t_k = T$), a confidence level $\gamma \in (0, 1)$, a tightness factor $\mu > 1$, a batch size b, and a distance function d. The output of the GoTube algorithm is a bounding tube that stochastically overapproximates at most by μ the propagated initial perturbation from the center x_0 with a probability higher than $1 - \gamma$.

GoTube starts by sampling a batch (tensor) $x^B \in \mathcal{B}_0^S$. It then iterates for the k steps of the time horizon T the following. After initializing the probability ensured to zero, and the visited states to the empty set, it loops until it reaches the desired confidence (probability) $1 - \gamma$, by increasingly taking additional batches. In each iteration, it integrates the center and the already available samples from their previous time step and the possibly new batches from their initial state (for simplicity, the pseudocode does not make this distinction explicit). GoTube then computes the maximum distance from the integrated samples to the integrated center, their local Lipschitz constant according to the variational equation of Eq. (1). Based on this information GoTube then computes the mean Lipschitz statistics and the cap radii accordingly. The total surface of the caps is then employed to compute and update the achieved confidence (probability). In Line 13, *computeProb* is a function of confidence level γ, cap radii r_x, systems dimension n and initial perturbation δ_0 and returns the probability $\Pr(\mu \cdot \bar{m} \geq m^\star)$. Once the desired confidence is achieved, GoTube exits the inner loop and computes the bounding ball in terms of its center and radius, which is given by tightness factor μ times the maximum distance $\bar{m}_{j,\mathcal{V}}$. After exiting the outer loop, GoTube returns the bounding tube.

Definition 3 (Lipschitz Cap) *Let $\mathcal{V}$ be the set of all sampled points, $x \in \mathcal{V}$ be a sample point on the surface of the initial ball, $\bar{m}_{j,\mathcal{V}} = \max_{x\in\mathcal{V}} d_j(x)$ be the sample maximum and $B(x, r_x)^S = B(x, r_x) \cap \mathcal{B}_0^S$ be a spherical cap around that point. We call the cap $B(x, r_x)^S$ a γ, t_j-Lipschitz cap, if it holds that* $\Pr\left(d_j(y) \leq \mu \cdot \bar{m}_{j,\mathcal{V}}\right) \geq 1-\gamma$ *for all $y \in B(x, r_x)^S$.*

Lipschitz caps around the samples are a stochastic version of local balls around samples, commonly used to cover state space. Intuitively, the points within a cap do not have to be explored. The difference with Lipschitz caps is, that we stochastically bound the values inside that space and develop a theory to enable us to calculate a probability of having found an upper bound of the true maximum $m_j^\star = d_j(x_j^\star) = \max_{x\in\mathcal{B}_0} d_j(x)$ of the optimization problem in Eq. (2). Our objective is to avoid the usage of interval arithmetic for computing the Lipschitz constant, as it impedes scaling up to continuous depth models. Instead, we define stochastic bounds on the Lipschitz constant to set the radius r_x of the Lipschitz caps, such that $\mu \cdot \bar{m}_{j,\mathcal{V}}$ is a γ-stochastic upper bound for all distances $d_j(y)$ at time t_j from values inside the ball $B(x, r_x)^S$.

Theorem 1 (Radius of Stochastic Lipschitz Caps) *Given a continuous-depth model f from Eq.* (1), *$\gamma \in (0,1)$, $\mu > 1$, target time t_j, the set of all sampled points $\mathcal{V}$, the number of sampled points $N = |\mathcal{V}|$, the sample maximum $\bar{m}_{j,\mathcal{V}} = \max_{x\in\mathcal{V}} d_j(x)$, the IVP solutions $\chi(t_j, x)$, and the corresponding stretching factors $\lambda_x = \|\partial_x \chi(t_j, x)\|$ for all $x \in \mathcal{V}$. Let us define $\hat{\gamma} = 1 - \sqrt{1-\gamma}$. Let $\Delta\lambda_\mathcal{V}$ be the $\sqrt{1-\gamma}$-quantile of a stochastic lower bound $F_{L,\hat{\gamma}}$ as defined by Lemma 1 in the Appendix:*

$$\Delta\lambda_\mathcal{V}(\gamma) = F_{L,\hat{\gamma}}^{-1}(\sqrt{1-\gamma}), \tag{3}$$

Let r_x be defined as:

$$r_x = \frac{\left(-\lambda_x + \sqrt{\lambda_x^2 + 4 \cdot \Delta\lambda_\mathcal{V} \cdot (\mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x))}\right)}{2 \cdot \Delta\lambda_\mathcal{V}}, \tag{4}$$

then it holds that:

$$\Pr\left(d_j(y) \leq \mu \cdot \bar{m}_{j,\mathcal{V}}\right) \geq 1-\gamma \quad \forall y \in B(x, r_x)^S, \tag{5}$$

and thus that $B(x, r_x)^S$ is a γ, t_j-Lipschitz cap.

The full proof is provided in the Appendix. *Proof sketch:* As $\Delta\lambda_\mathcal{V}$ is the $\sqrt{1-\gamma}$-quantile of $\max_{x,y} |\lambda_x - \lambda_y| / \|x - y\|$, it holds that $\Pr(\lambda_y \leq \lambda_x + \Delta\lambda_\mathcal{V} \cdot \|x-y\|) \geq 1-\gamma$. Therefore Eq. (4) follows by solving the following equation: $(\mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x)) = \lambda_x r_x + \Delta\lambda_\mathcal{V} r_x^2$.

Using conditional probability, we are able to state that the convergence guarantee holds for the GoTube Algorithm, thus ensuring that the Algorithm terminates in finite time even using stochastic Lipschitz caps around the samples instead of deterministic local balls.

Theorem 2 (Convergence via Lipschitz Caps) *Given the tightness factor $\mu > 1$, the set of all sampled points $\mathcal{V}$ and the sample maximum $\bar{m}_{j,\mathcal{V}} = \max_{x\in\mathcal{V}} d_j(x)$. Let the initial ball maximum be defined by $m_j^\star = \max_{x\in\mathcal{B}_0} d_j(x)$. Then:*

$$\forall\gamma \in (0,1), \exists N \in \mathbb{N} \textit{ s.t. } \Pr(\mu \cdot \bar{m}_{j,\mathcal{V}} \geq m_j^\star) \geq 1-\gamma \tag{6}$$

where $N = |\mathcal{V}|$ is the number of sampled points.

The full proof is provided in the Appendix. *Proof sketch:* Let $x_j^\star$ be a point such that $d_j(x_j^\star) = m_j^\star$. Given $\gamma \in (0,1)$ and cap radii r_x, we expand the convergence guarantee from deterministic local balls to stochastic Lipschitz caps. For local balls it holds that $\exists N \in \mathbb{N}$: $\Pr(\exists x \in \mathcal{V}\colon B(x, r_x)^S \ni x_j^\star) \geq \sqrt{1-\gamma}$. Using a set of sampled points $\mathcal{V}$ with cardinality N and using $1 - \sqrt{1-\gamma}$ instead of γ in Eq. (3) and Theorem 1, the resulting probability is larger than $\sqrt{1-\gamma}$. From the definition of a Lipschitz cap it follows that $\Pr(d_j(x^\star) \leq \mu \cdot \bar{m}_{j,\mathcal{V}} | \exists x \in \mathcal{V}\colon B(x, r_x)^S \ni x^\star) \geq \sqrt{1-\gamma}$. For any sets A, B it holds that $\Pr(A) \geq \Pr(A \cap B) = \Pr(A|B) \cdot \Pr(B)$, thus we multiply both probabilities and therefore Eq. (6) holds.

Experimental Evaluation

We perform a diverse set of experiments with GoTube to evaluate its performance and identify its characteristics and limits in verifying continuous-time systems with increasing complexity. We run our evaluations on a standard workstation machine setup (12 vCPUs, 64GB memory) equipped with a single GPU for a per-run timeout of 1 hour (except for runtimes reported in Figure 4). We perform experiments on 5 neural models: 2 neural models in Table 2 and Table 4 (CartPole-v1+LTC, CartPole-v1+CTRNN) and 3 additional neural models in Figure 4 (LDS, Pendulum and Oscillatory controlled by CT-RNN), where the Oscillatory benchmark model is controlled by a CT-RNN, that is twice as complex as the ones that LRT-NG can handle.

On the volume of the bounding balls with GoTube

Our first experimental evaluation concerns the overapproximation errors of the constructed bounding tubes. An ideal reachability tool should be able to output an as tight as possible tube that encloses the system's executions. Consequently, as our comparison metric, we will report the average volume of the bounding balls, with less volume is better. We use the benchmarks and settings of (Gruenbacher et al. 2020) (same radii, time horizons, and models) as the

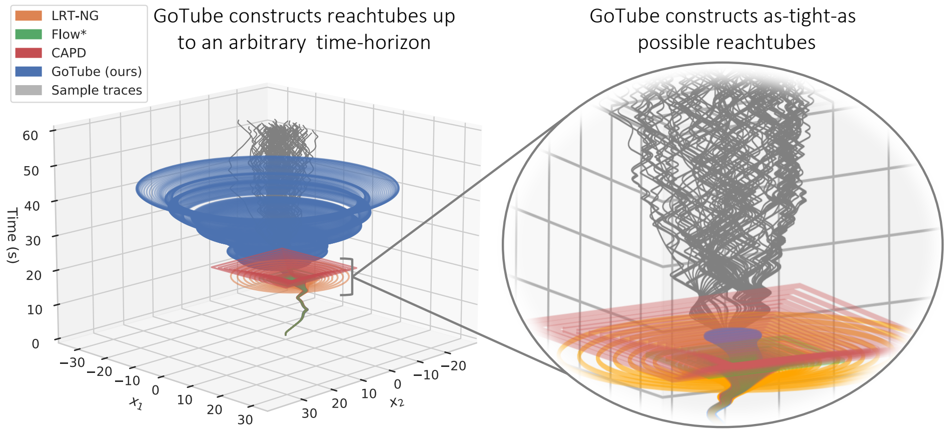

Figure 3: Visualization of the reachtubes constructed for the Dubin's car model with various reachability methods. While the tubes computed by existing methods (LRT-NG, Flow* and CAPD) explode at $t \approx 20s$ (this moment is shown on the right side of the figure) due to the accumulation of overapproximation errors (the infamous wrapping effect), GoTube can keep tight bounds beyond $t > 40s$ for a 99% confidence level (using 20000 samples, $\mu = 1.1$ and runtime of one hour). Note also the chaotic nature of 100 executions.

basis of our evaluation. In particular, we compare GoTube to the deterministic, state-of-the-art reachability tools LRT-NG, Flow*, CAPD, and LRT. We measure the volume of GoTube's balls at the confidence levels of 90% and 99%, using $\mu = 1.1$ as the tightness factor (in the third experiment we will talk in more detail about the trade-off between tightness and runtime).

The results are shown in Table 2. For the first five benchmarks, which are classical dynamical systems, we use the small time horizons T and small initial radii δ_0, which the other tools could handle. GoTube, with 99% confidence, achieves a competitive performance to the other tools, coming out on top in 3 out of 5 benchmarks - using $\mu = 1.1$ as the tightness bound. Intuitively this means, we are confident that the overapproximation includes all executions with a confidence level $1 - \lambda$, but this overapproximation might not be as tight as desired. GoTube is able to achieve any desired tightness by reducing μ and increasing the runtime. The goal of our paper is to improve Benchmarks with perturbation radii, time horizons and model size (neural network policies) that other tools are not able to handle. We did additional experiments exemplary for the Robotarm benchmark. Table 3 shows that if we compute the average volume either for a longer time horizon or for a bigger initial perturbation, GoTube performs better than LRT-NG, which is the best conservative tool for Robotarm. GoTube is doing even better after a very short initial phase or for a bigger initial perturbation for a reasonable small computation time of less than 1 hour.

The specific reachtubes and the chaotic nature of hundred executions of Dubin's car are shown in Figure 3. As one can see, the GoTube reachtube extends to a much longer time horizon, which we fixed at 40s. All other tools blew up before 20s. For the two problems involving neural networks, GoTube produces significantly tighter reachtubes.

Table 2: Comparison of GoTube (using tightness bound $\mu = 1.1$) to existing reachability methods. The first five benchmarks concern classical dynamical systems, whereas the two bottom rows correspond to time-continuous RNN models (LTC= liquid time-constant networks) in a closed feedback loop with an RL environment (Hasani et al. 2021; Vorbach et al. 2021). The numbers show the volume of the constructed tube. Lower is better; best number in bold.

Benchmark	LRT-NG	Flow*	CAPD	LRT	GoTube	
					(90%)	(99%)
Brusselator	1.5e-4	9.8e-5	3.6e-4	6.1e-4	8.6e-5	**8.6e-5**
Van Der Pol	4.2e-4	**3.5e-4**	1.5e-3	3.5e-4	3.5e-4	**3.5e-4**
Robotarm	**7.9e-11**	8.7e-10	1.1e-9	Fail	2.5e-10	2.5e-10
Dubins Car	0.131	4.5e-2	0.1181	385	2.5e-2	**2.6e-2**
Cardiac Cell	**3.7e-9**	1.5e-8	4.4e-8	3.2e-8	4.2e-8	4.3e-8
CartPole-v1+LTC	4.49e-33	Fail	Fail	Fail	2.6e-37	**4.9e-37**
CartPole-v1+CTRNN	3.9e-27	Fail	Fail	Fail	9.9e-34	**1.2e-33**

Table 3: The numbers show the average volume of the constructed tube calculated by LRT-NG and GoTube over different time horizons and initial perturbations for the Robotarm benchmark.

Benchmark	time horizon	initial perturbation	LRT-NG	GoTube (99%)
Robotarm	0s-3.5s	0.005	**8.8e-10**	3.1e-9
	3.5s-40s	0.005	3.1e-12	**2.7e-12**
	10s-40s	0.005	1.9e-15	**3.1e-17**
	0s-2.5s	0.05	9.4e-3	**4.1e-5**
	0s-0.1s	0.5	2.26	**0.45**

GoTube provides safety bounds up an arbitrary time horizon

In our second experiment, we evaluate for how long GoTube and existing methods can construct a reachtube before exploding due to overapproximation errors. To do so, we extend the benchmark setup by increasing the time horizon for which the tube should be constructed, use tightness bound

$\mu = 1.1$ and set a 95% confidence level, that is, probability of being conservative.

We have extended the time horizon from 1 second to 10 seconds in the evaluation of Table 4 and Figure 1 and from 20 seconds to 40 seconds in Figure 3. Our theory allows an arbitrary time-horizon, as our experiments show us that after computing the bounding tube for time t_1, every timestep needs roughly the same computation time, so there is a linear growth of computation time when increasing the time horizon.

Table 4: Results of the extended benchmark by longer time horizons. The numbers show the volume of the constructed tube, "Blowup" indicates that the method produced `Inf` or `NaN` values due to a blowup. Lower is better; the best method is shown in bold.

Benchmark	CartPole-v1+CTRNN		CartPole-v1+LTC	
Time horizon	1s	10s	0.35s	10s
LRT	Blowup	Blowup	Blowup	Blowup
CAPD	Blowup	Blowup	Blowup	Blowup
Flow*	Blowup	Blowup	Blowup	Blowup
LRT-NG	3.9e-27	Blowup	4.5e-33	Blowup
GoTube (ours)	**8.8e-34**	**1.1e-19**	**4.9e-37**	**8.7e-21**

The results in Table 4 demonstrate that GoTube produces significantly longer reachtubes than all considered state-of-the-art approaches, without suffering from severe overapproximation errors. Particularly, Figure 1 visualizes the difference to the existing methods and overapproximation margins for two example dimensions of the CartPole-v1 environment and its CT-RNN controller.

GoTube can trade runtime for reachtube tightness

In our last experiment, we introduced a new set of benchmark models entirely based on continuous-time recurrent neural networks. The first model is an unstable linear dynamical system of the form $\dot{x} = Ax + Bu$ that is stabilized by a CT-RNN policy via actions u. The second model corresponds to the inverted pendulum environment, which is similar to the CartPole environment but differs in that the control actions are applied via a torque vector on the pendulum directly instead of moving a cart. The CT-RNN policies for these two environments were trained using deep RL. Our third new benchmark model concerns the analysis of the learned dynamics of a CT-RNN trained on supervised data. In particular, by using the reachability frameworks, we aim to assess if the learned network expressed oscillatory behavior. The CT-RNN state vector consists of 16 dimensions, which is twice as much as existing CT-RNN reachability benchmarks (Gruenbacher et al. 2020).

Here, we study how GoTube can trade runtime for the volume of the constructed reachtube through its tightness factor μ. In particular, we run GoTube on our newly proposed benchmark with various values of μ. We then plot GoTube's runtime (x-axis) and volume size (y-axis) as a function of μ. The resulting curves show the Pareto-front of runtime-volume trade-off achievable with GoTube.

Figure 4 shows the results for a time horizon of 10s in the first two examples, and of 2s in the last example. The Oscillatory benchmark model is controlled by a CT-RNN, that is already twice as complex as the ones that LRT-NG can handle. The corresponding volumes obtained with the lowest μ are: 4.3e-13, 2.4e-12, and 2.1e-3. All other tools could not handle these models. Our results demonstrate that GoTube can adapt to different runtime and tightness constraints and set a new benchmark for future methods to compare with.

Discussions, Scope and Conclusions

We proposed GoTube, a new stochastic verification algorithm that provides robustness guarantees (also safety guarantees if a set of states to be avoided is given) for high-dimensional, time-continuous systems. GoTube is stable and sets the state-of-the-art in terms of its ability to scale to time horizons well beyond what has been previously possible. It also allows a larger perturbation radius for the initial ball, for which other verification methods fail. Lastly, GoTube's scalability enables it to readily handle the verification of advanced continuous-depth neural models, a setting where state-of-the-art deterministic approaches fail.

SLR versus GoTube? SLR combines symbolic with statistical reachability techniques. However, no implementation is available to date. For comparison purposes, we implemented SLR on our own and observed that while it does not blow up in space, it blows up in time. As a consequence, we were not able to use SLR to construct reachtubes for our high-dimensional benchmarks.

Sample blow up in GoTube? As a pure Monte-Carlo technique, the number of samples N to be taken depends on both the confidence coefficient λ and the tightness coefficient μ as well as on the system's dimensionality. As a consequence, for very small values of these coefficients, the number of samples tends to blow up. The goal of symbolic techniques is exactly the one to avoid such a blowup. However, in our experiments, we observed that GoTube outperformed in all cases the symbolic techniques. This implies that the overapproximation error of symbolic techniques is more problematic than the blowup in the number of samples for a large number of dimensions.

What about Gaussian Processes? Gaussian Processes (GPs) are powerful stochastic models which can be used for stochastic reachability analysis (Bortolussi and Sanguinetti 2014) and uncertainty estimation for stochastic dynamical systems (Gal 2016). The major shortcoming of GPs is that they simply cannot scale to the complex continuous-time systems that we tested here. Moreover, Gaussian Processes have a large number of hyperparameters, which can be challenging to tune across different benchmarks.

Limitations of GoTube. GoTube does not necessarily perform better in terms of average volume of the bounding balls for smaller tasks and shorter time horizons if not choosing a very small μ, as shown in Table 2. GoTube is not yet suitable for the verification of stochastic dynamical systems, for instance, Neural Stochastic Differential Equations (Neural SDEs) (Li et al. 2020; Xu et al. 2021). Although GoTube is considerably more computationally efficient than existing methods, the dimensionality of the system, as well as the

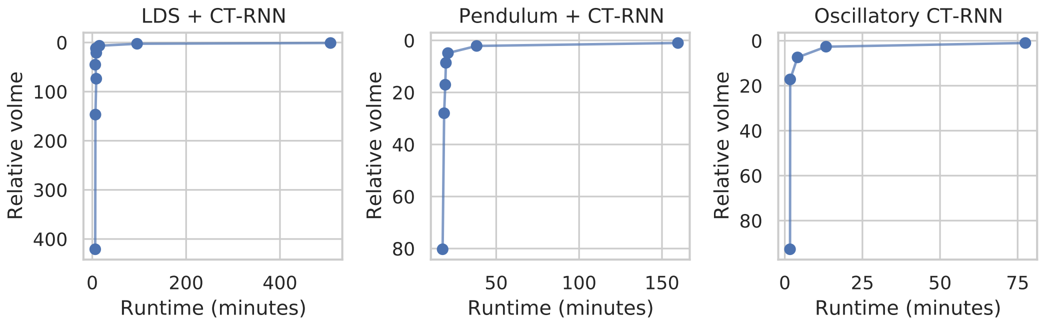

Figure 4: GoTube's runtime (x-axis) and volume size (y-axis) as a function of the tightness factor μ. Volume was normalized by the volume obtained with the lowest μ (4.3e-13, 2.4e-12, and 2.1e-38 in particular).

type of numerical ODE solver exponentially, affect their performance. We can improve on this limitation by using Hypersolvers (Poli et al. 2020), closed-form continuous depth models, and compressed representations of neural ODEs.

Future of GoTube. GoTube opens many avenues for future research. The most straightforward next step is to search for better intermediate steps in Algorithm 1. For instance, better ways to compute the Lipschitz constant and to improve the sampling process. GoTube is now applicable for complex deterministic ODE systems; it would be an important line of work to find ways to marry reachability analysis with machine learning approaches to verify neural SDEs as well. Last but not least, we believe that there is a close relationship between stochastic reachability analysis and uncertainty estimation techniques used for deep learning models (Abdar et al. 2021). Uncertainty-aware verification could be worth exploring based on what we learned with GoTube.

Acknowledgements

SG is funded by the Austrian Science Fund (FWF) project number W1255-N23. ML and TH are supported in part by FWF under grant Z211-N23 (Wittgenstein Award) and the ERC-2020-AdG 101020093. SS is supported by NSF awards DCL-2040599, CCF-1918225, and CPS-1446832. RH and DR are partially supported by Boeing. RG is partially supported by Horizon-2020 ECSEL Project grant No. 783163 (iDev40).

References

Abdar, M.; Pourpanah, F.; Hussain, S.; Rezazadegan, D.; Liu, L.; Ghavamzadeh, M.; Fieguth, P.; Cao, X.; Khosravi, A.; Acharya, U. R.; et al. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 274–283. PMLR.

Bak, S.; Tran, H.-D.; Hobbs, K.; and Johnson, T. T. 2020. Improved geometric path enumeration for verifying ReLU neural networks. In *CAV*, 66–96. Springer.

Bhowmick, A.; D'Souza, M.; and Raghavan, G. S. 2021. LipBaB: Computing exact Lipschitz constant of ReLU networks. *arXiv preprint arXiv:2105.05495*.

Bortolussi, L.; and Sanguinetti, G. 2014. A Statistical Approach for Computing Reachability of Non-linear and Stochastic Dynamical Systems. In Norman, G.; and Sanders, W., eds., *Quantitative Evaluation of Systems*, 41–56. Cham: Springer International Publishing.

Bunel, R.; De Palma, A.; Desmaison, A.; Dvijotham, K.; Kohli, P.; Torr, P.; and Kumar, M. P. 2020a. Lagrangian decomposition for neural network verification. In *UAI*, 370–379. PMLR.

Bunel, R.; Mudigonda, P.; Turkaslan, I.; Torr, P.; Lu, J.; and Kohli, P. 2020b. Branch and bound for piecewise linear neural network verification. *JMLR*, 21(2020).

Bunel, R. R.; Turkaslan, I.; Torr, P.; Kohli, P.; and Mudigonda, P. K. 2018. A Unified View of Piecewise Linear Neural Network Verification. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS*, volume 31. Curran Associates, Inc.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS 31*, 6571–6583. Curran Associates, Inc.

Chen, X.; Ábrahám, E.; and Sankaranarayanan, S. 2013. Flow*: an Analyzer for Non-linear Hybrid Systems. In *CAV*, 258–263.

Cyranka, J.; Islam, M. A.; Byrne, G.; Jones, P.; Smolka, S. A.; and Grosu, R. 2017. Lagrangian Reachabililty. In Majumdar, R.; and Kunčak, V., eds., *CAV*, 379–400. Heidelberg, Germany: Springer.

De Palma, A.; Bunel, R.; Desmaison, A.; Dvijotham, K.; Kohli, P.; Torr, P. H.; and Kumar, M. P. 2021. Improved Branch and Bound for Neural Network Verification via Lagrangian Decomposition. *arXiv preprint arXiv:2104.06718*.

Devonport, A.; Khaled, M.; Arcak, M.; and Zamani, M. 2020. PIRK: Scalable Interval Reachability Analysis for High-Dimensional Nonlinear Systems. In Lahiri, S. K.;

and Wang, C., eds., *Computer Aided Verification*, 556–568. Cham: Springer International Publishing.

Donzé, A. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, 167–170. Edinburgh, UK: Springer.

Donzé, A.; and Maler, O. 2007. Systematic simulation using sensitivity analysis. In *HSCC*, 174–189.

Duggirala, P. S.; Mitra, S.; Viswanathan, M.; and Potok, M. 2015. C2E2: A Verification Tool for Stateflow Models. In Baier, C.; and Tinelli, C., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 68–82. Berlin, Heidelberg: Springer Berlin Heidelberg.

Dvoretzky, A.; Kiefer, J.; and Wolfowitz, J. 1956. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *Annals of Mathematical Statistics*, 27(3): 642–669.

Ehlers, R. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 269–286. Springer.

Enszer, J. A.; and Stadtherr, M. A. 2011. Verified Solution and Propagation of Uncertainty in Physiological Models. *Reliab. Comput.*, 15(3): 168–178.

Fan, C.; Kapinski, J.; Jin, X.; and Mitra, S. 2017. Simulation-Driven Reachability Using Matrix Measures. *ACM Trans. Embed. Comput. Syst.*, 17(1).

Fazlyab, M.; Robey, A.; Hassani, H.; Morari, M.; and Pappas, G. 2019. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *NeurIPS*, volume 32. Curran Associates, Inc.

Fränzle, M.; Hahn, E.; Hermanns, H.; Wolovick, N.; and Zhang, L. 2011. Measurability and safety verification for stochastic hybrid systems. In *HSCC*, 43–52.

Gal, Y. 2016. Uncertainty in deep learning. *University of Cambridge*, 1(3): 4.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. Satisfiability modulo ODEs. In *2013 Formal Methods in Computer-Aided Design*, 105–112.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2018. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*.

Gruenbacher, S.; Cyranka, J.; Lechner, M.; Islam, M. A.; Smolka, S. A.; and Grosu, R. 2020. Lagrangian Reachtubes: The Next Generation. In *CDC*, 1556–1563.

Gruenbacher, S.; Hasani, R.; Lechner, M.; Cyranka, J.; Smolka, S. A.; and Grosu, R. 2021. On the Verification of Neural ODEs with Stochastic Guarantees. *AAAI*, 35(13): 11525–11535.

Gurung, A.; Ray, R.; Bartocci, E.; Bogomolov, S.; and Grosu, R. 2019. Parallel reachability analysis of hybrid systems in xspeed. *International Journal on Software Tools for Technology Transfer*, 21(4): 401–423.

Hansen, E.; and Walster, G. W. 2003. *Global optimization using interval analysis: revised and expanded*, volume 264. CRC Press.

Hansen, N.; and Ostermeier, A. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2): 159–195.

Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; and Grosu, R. 2021. Liquid Time-constant Networks. *AAAI*, 35(9).

Henriksen, P.; and Lomuscio, A. 2020. Efficient neural network verification via adaptive refinement and adversarial search. In *ECAI 2020*, 2513–2520. IOS Press.

Henzinger, T. A.; Lechner, M.; and Zikelic, D. 2021. Scalable Verification of Quantized Neural Networks. In *AAAI*, volume 35, 3787–3795.

Huang, C.; Chen, X.; Lin, W.; Yang, Z.; and Li, X. 2017. Probabilistic Safety Verification of Stochastic Hybrid Systems Using Barrier Certificates. *ACM Trans. Embed. Comput. Syst.*, 16(5s).

Igel, C.; Hansen, N.; and Roth, S. 2007. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1): 1–28.

Immler, F. 2015. Verified Reachability Analysis of Continuous Systems. In Baier, C.; and Tinelli, C., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 37–51. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kapela, T.; Mrozek, M.; Wilczak, D.; and Zgliczynski, P. 2020. CAPD:: DynSys: a flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Pre-Print - ww2.ii.uj.edu.pl*.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*, 97–117. Springer.

Kvasov, D. E.; and Sergeyev, Y. D. 2013. Lipschitz global optimization methods in control problems. *Automation and Remote Control*, 74(9): 1435–1448.

Lechner, M.; Hasani, R.; Amini, A.; Henzinger, T. A.; Rus, D.; and Grosu, R. 2020. Neural circuit policies enabling auditable autonomy. *Nature MI*, 2(10): 642–652.

Lechner, M.; Hasani, R.; Grosu, R.; Rus, D.; and Henzinger, T. A. 2021. Adversarial Training is Not Ready for Robot Learning. *arXiv preprint arXiv:2103.08187*.

Li, D.; Bak, S.; and Bogomolov, S. 2020. Reachability Analysis of Nonlinear Systems Using Hybridization and Dynamics Scaling. In Bertrand, N.; and Jansen, N., eds., *Formal Modeling and Analysis of Timed Systems*, 265–282. Cham: Springer International Publishing.

Li, X.; Wong, T.-K. L.; Chen, R. T.; and Duvenaud, D. 2020. Scalable gradients for stochastic differential equations. In *AISTATS*, 3870–3882. PMLR.

Lu, J.; and Mudigonda, P. 2020. Nueral network branching for nueral network verification. In *ICLR 2020*. Open Review.

Malherbe, C.; and Vayatis, N. 2017. Global Optimization of Lipschitz Functions. In *Proceedings of the 34th ICML - Volume 70*, ICML'17, 2314–2323. JMLR.org.

Massart, P. 1990. The tight constant in the Dvoretzky–Kiefer–Wolfowitz inequality. *Annals of Probability*, 18(3): 1269–1283.

Meyer, P.-J.; Devonport, A.; and Arcak, M. 2019. TIRA: Toolbox for Interval Reachability Analysis. In *Association for Computing Machinery*, HSCC '19, 224–229. New York, NY, USA.

Mirman, M.; Gehr, T.; and Vechev, M. 2018. Differentiable abstract interpretation for provably robust neural networks. In *ICML*, 3578–3586. PMLR.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Neumaier, A. 2004. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13: 271–369.

Poli, M.; Massaroli, S.; Yamashita, A.; Asama, H.; Park, J.; et al. 2020. Hypersolvers: Toward Fast Continuous-Depth Models. *NeurIPS*, 33.

Pontryagin, L. S. 2018. *Mathematical theory of optimal processes*. Routledge.

Salman, H.; Yang, G.; Zhang, H.; Hsieh, C.-J.; and Zhang, P. 2019. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *NeurIPS*, volume 32. Curran Associates, Inc.

Shmarov, F.; and Zuliani, P. 2015a. ProbReach: A Tool for Guaranteed Reachability Analysis of Stochastic Hybrid Systems. In Bogomolov, S.; and Tiwari, A., eds., *SNR-CAV*, volume 37, 40–48.

Shmarov, F.; and Zuliani, P. 2015b. ProbReach: verified probabilistic delta-reachability for stochastic hybrid systems. In *HSCC*, 134–139. ACM.

Singh, G.; Maurer, J.; Müller, C.; Mirman, M.; Gehr, T.; Hoffmann, A.; Tsankov, P.; Cohen, D. D.; Püschel, M.; and Vechev, M. 2020. ETH robustness analyzer for neural networks (ERAN). *URL https://github. com/eth-sri/eran*.

Tjandraatmadja, C.; Anderson, R.; Huchette, J.; Ma, W.; PATEL, K. K.; and Vielma, J. P. 2020. The Convex Relaxation Barrier, Revisited: Tightened Single-Neuron Relaxations for Neural Network Verification. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *NeurIPS*, volume 33, 21675–21686. Curran Associates, Inc.

Vinod, A. P.; and Oishi, M. M. 2021. Stochastic reachability of a target tube. *Automatica*, 125: 109458.

Vorbach, C.; Hasani, R.; Amini, A.; Lechner, M.; and Rus, D. 2021. Causal Navigation by Continuous-time Neural Networks. *arXiv preprint arXiv:2106.08314*.

Wang, Q.; Zuliani, P.; Kong, S.; Gao, S.; and Clarke, E. M. 2015. SReach: A Probabilistic Bounded Delta-Reachability Analyzer for Stochastic Hybrid Systems. In Roux, O.; and Bourdon, J., eds., *Computational Methods in Systems Biology*, 15–27. Cham: Springer International Publishing.

Xu, W.; Chen, R. T.; Li, X.; and Duvenaud, D. 2021. Infinitely Deep Bayesian Neural Networks with Stochastic Differential Equations. *arXiv preprint arXiv:2102.06559*.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS*, volume 31. Curran Associates, Inc.

Zhigljavsky, A.; and Zilinskas, A. 2008. *Stochastic Global Optimization*, volume 9 of *Springer Optimization and Its Applications*. Springer US.

Appendix

Proofs of the Theorems

Lemma 1 (Stochastic lower bound $F_{L,\gamma}$) *Consider the experiment of randomly sampling two times m points of the initial ball $\mathcal{B}_0$: $(a_1,\dots,a_m)$ and $(b_1,\dots,b_m)$. Let $g : \mathbb{R}^n \to \mathbb{R}$ be a real-valued function and $X = \max_{i=1}^{m} |g(a_i) - g(b_i)|/\|a_i - b_i\|$ be a random variable with the unknown cumulative distribution function F. Let $(x_1,\dots,x_n) \sim X$ be independent, identically distributed samples with the empirical distribution function $\hat{F}_n(x) = \sum_{i=1}^{n} \mathbb{1}_{x_i \le x}$. Let G_n be a generalized extreme value distribution fitted to the empirical distribution function $\hat{F}_n$ and let D_n^- describe the goodness of fit, being the one-sided Kolmogorov–Smirnov statistic:*

$$D_n^- = \sup_x (G_n(x) - \hat{F}_n(x)) \tag{S1}$$

Given the confidence level γ and $\alpha = \min(\gamma, 0.5)$, then let us define $\epsilon_{n,\gamma}$ and $F_{L,\gamma}$ as follows:

$$\epsilon_{n,\gamma} = \sqrt{\frac{\ln\frac{1}{\alpha}}{2n}} \tag{S2}$$

$$F_{L,\gamma}(x) = G_n(x) - \epsilon_{n,\gamma} - D_n^- \tag{S3}$$

Then it holds that:

$$Pr(\sup_x (F_{L,\gamma}(x) - F(x)) \le 0) \ge 1 - \gamma, \tag{S4}$$

which intuitively means that $F_{L,\gamma}$ is a lower bound of F with confidence γ.

Proof. The Fisher-Tippett-Gnedenko theorem states that the distribution of a normalized maximum converges to the generalized extreme value distribution, if the distribution of the normalized maximum does converge. So intuitively that theorem is similar to the central limit theorem for the averages, but for the normalized maxima. Consequently, we start by fitting the empirical distribution function $\hat{F}_n$ by a generalized extreme value distribution G_n and compute Eq. (S1).

The Dvoretzky-Kiefer-Wolfowitz inequality (Dvoretzky, Kiefer, and Wolfowitz 1956) with a tight constant determined by (Massart 1990), states that for all $\epsilon \ge \sqrt{\frac{1}{2n}\ln 2}$, it holds that:

$$Pr(\sup_x (\hat{F}_n(x) - F(x)) > \epsilon) \le e^{-2n\epsilon^2} \tag{S5}$$

Solving $\gamma = e^{-2n\epsilon^2}$ for ϵ and considering Massarts lower bound for ϵ, yields:

$$Pr(\sup_x (\hat{F}_n(x) - F(x)) > \epsilon_{n,\gamma}) \le \gamma, \tag{S6}$$

with $\epsilon_{n,\gamma}$ as defined in Eq. (S2). We use the triangular inequality for supremum and the monotony of the probability measure as follows:

$$Pr\Big(\sup_x \big(G_n(x) - F(x)\big) > \epsilon_{n,\gamma} + D_n^-\Big) = \tag{S7}$$

$$= Pr\Big(\sup_x \big(G_n(x) - \hat{F}_n(x) + \tag{S8}$$

$$+ \hat{F}_n(x) - F(x)\big) > \epsilon_{n,\gamma} + D_n^-\Big) \tag{S9}$$

$$\le Pr\Big(\sup_x \big(G_n(x) - \hat{F}_n(x)\big) + \tag{S10}$$

$$+ \sup_x \big(\hat{F}_n(x) - F(x)\big) > \epsilon_{n,\gamma} + D_n^-\Big) \tag{S11}$$

$$\overset{\text{(S1)}}{=} Pr\Big(\sup_x \big(\hat{F}_n(x) - F(x)\big) > \epsilon_{n,\gamma}\Big) \tag{S12}$$

$$\overset{\text{(S6)}}{\le} \gamma, \tag{S13}$$

from which it follows directly, that Eq. (S4) hold.

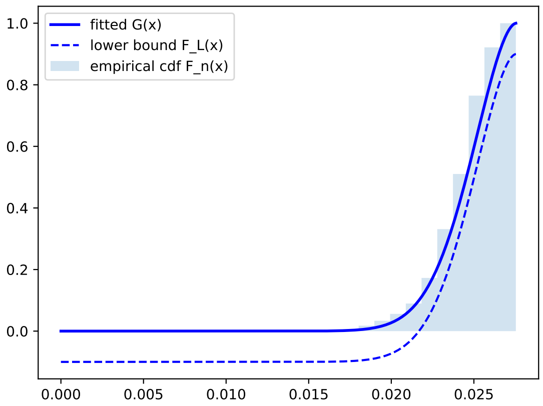

Figure S1: Visualisation of the stochastic lower bound $F_{L,\gamma}$ of Lemma 1.

Theorem 1 (Radius of Stochastic Lipschitz Caps) *Given a continuous-depth model f from Eq. (1) in the main paper ($\partial_t x = f(x)$ with $x(t_0) \in B(x_0, \delta_0)$), $\gamma \in (0,1)$, $\mu > 1$, target time t_j, the set of all sampled points $\mathcal{V}$, the number of sampled points $N = |\mathcal{V}|$, the sample maximum $\bar{m}_{j,\mathcal{V}} = \max_{x \in \mathcal{V}} d_j(x)$, the IVP solutions $\chi(t_j, x)$, and the corresponding stretching factors $\lambda_x = \|\partial_x \chi(t_j, x)\|$ for all $x \in \mathcal{V}$. Let us define $\hat{\gamma} = 1 - \sqrt{1-\gamma}$. Let $\Delta\lambda_{\mathcal{V}}$ be the $\sqrt{1-\gamma}$-quantile of a stochastic lower bound $F_{L,\hat{\gamma}}$ as defined in Eq.* (S3) *of Lemma 1:*

$$\Delta\lambda_{\mathcal{V}}(\gamma) = F_{L,\hat{\gamma}}^{-1}(\sqrt{1-\gamma}), \tag{S14}$$

Let r_x be defined as:

$$r_x = \frac{\left(-\lambda_x + \sqrt{\lambda_x^2 + 4 \cdot \Delta\lambda_{\mathcal{V}} \cdot (\mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x))}\right)}{2 \cdot \Delta\lambda_{\mathcal{V}}}, \tag{S15}$$

then it holds that:

$$\Pr\left(d_j(y) \le \mu \cdot \bar{m}_{j,\mathcal{V}}\right) \ge 1 - \gamma \quad \forall y \in B(x, r_x)^S, \tag{S16}$$

and thus that $B(x, r_x)^S$ is a γ, t_j-Lipschitz cap.

Proof. Let $\{x_1, \dots, x_n\}$ be n independent experiments by sampling from $X = \max_{i=1}^{m} |\lambda_{a_i} - \lambda_{b_i}|/\|a_i - b_i\|$ as defined in Lemma 1, where each variable is the maximum of m executions. From Eq. (S4) it follows that:

$$Pr(F_{L,\hat{\gamma}(x)} \leq F(x)) \geq 1 - \hat{\gamma} = \sqrt{1-\gamma} \tag{S17}$$

Let us now derive the probability of X being less or equal to $\Delta\lambda_{\mathcal{V}}$ defined by Eq. (S14). For any sets A, B it holds that $\Pr(A) \geq \Pr(A \cap B) = \Pr(A|B) \cdot \Pr(B)$, thus:

$$\begin{aligned} &Pr(X \leq \Delta\lambda_{\mathcal{V}}) \geq && \text{(S18)} \\ &\geq Pr\Big(X \leq \Delta\lambda_{\mathcal{V}} | F_{L,\hat{\gamma}(x)} \leq F(x)\Big) \cdot && \text{(S19)} \\ &\qquad \cdot Pr\Big(F_{L,\hat{\gamma}(x)} \leq F(x)\Big) && \text{(S20)} \end{aligned}$$

Let us have a look on Eq. (S19): As $Pr(X \leq \Delta\lambda) = F(\Delta\lambda)$ and we are looking for the conditional probability depending on $F_{L,\hat{\gamma}(x)} \leq F(x)$, we can use $F_{L,\hat{\gamma}}(\Delta\lambda)$ as a lower bound of Eq. (S19) and thus, using Eq. (S17):

$$Pr(X \leq \Delta\lambda_{\mathcal{V}}) \geq F_{L,\hat{\gamma}}(\Delta\lambda_{\mathcal{V}}) \cdot \sqrt{1-\gamma} \tag{S21}$$

As Eq. (S14) defines $\Delta\lambda_{\mathcal{V}}$ as the $\sqrt{1-\gamma}$-quantile of $F_{L,\gamma}$, we can further state that

$$\begin{aligned} &\Pr(X \leq \Delta\lambda_{\mathcal{V}}) \geq 1 - \gamma, \\ &\text{with} \quad X = \max_{i=1}^{m} \left[\frac{|\lambda_{a_i} - \lambda_{b_i}|}{\|a_i - b_i\|} \right] \end{aligned} \tag{S22}$$

Let $Y = |\lambda_a - \lambda_b|/\|a - b\|$ be another random variable with a, b being to sample points of the initial ball $\mathcal{B}_0$. From Eq. (S22) it holds that

$$Pr(Y \leq \Delta\lambda_{\mathcal{V}}) \geq Pr(X \leq \Delta\lambda_{\mathcal{V}}) \geq 1 - \gamma \tag{S23}$$

It trivially holds for $x, y \in \mathcal{V}$ that:

$$\begin{aligned} \lambda_y &= \lambda_x + \frac{\lambda_y - \lambda_x}{\|x - y\|} \cdot \|x - y\| \\ &\leq \lambda_x + \frac{|\lambda_x - \lambda_y|}{\|x - y\|} \cdot \|x - y\| \end{aligned} \tag{S24}$$

From Eq. (S23) it follows that:

$$\begin{aligned} &Pr\Bigg(\lambda_x + \frac{|\lambda_x - \lambda_y|}{\|x - y\|} \cdot \|x - y\| \leq && \text{(S25)} \\ &\qquad \leq \lambda_x + \Delta\lambda_{\mathcal{V}} \cdot \|x - y\|\Bigg) \geq 1 - \gamma && \text{(S26)} \end{aligned}$$

and using the monotony of the probability measure:

$$\Pr\left(\lambda_y \leq \lambda_x + \Delta\lambda_{\mathcal{V}} \cdot \|x - y\|\right) \geq 1 - \gamma \tag{S27}$$

Using the mean value inequality for vector-valued functions it holds that:

$$\begin{aligned} |d_j(x) - d_j(y)| &= |\, \|\chi(t_j, x) - \chi(t_j, x_0)\| - \\ &\quad - \|\chi(t_j, y) - \chi(t_j, x_0)\| \,| \quad \{\text{triangle inequality}\} \\ &\leq \|\chi(t_j, x) - \chi(t_j, y)\| \quad \{\text{mean value theorem}\} \\ \Rightarrow \exists z \in [x, y]&\colon |d_j(x) - d_j(y)| \\ &\leq \|\partial_x \chi(t_j, z)\| \|x - y\| = \lambda_z \cdot \|x - y\| \end{aligned}$$

Combining this with Eq. (S27) and thus using $\lambda_x + \Delta\lambda_{\mathcal{V}} \cdot \|x - y\|$ as a probabilistic upper bound for λ_z, we obtain the following results for all y with $\|x - y\| \leq r_x$:

$$\begin{aligned} &\Pr\big(|d_j(x) - d_j(y)| \leq \\ &\quad \leq (\lambda_x + \Delta\lambda_{\mathcal{V}} \cdot \|x - y\|) \cdot \|x - y\|\big) \geq 1 - \gamma \\ &\Pr\big(|d_j(x) - d_j(y)| \leq \\ &\quad \leq (\lambda_x + \Delta\lambda_{\mathcal{V}} \cdot r_x) \cdot r_x\big) \geq 1 - \gamma \end{aligned} \tag{S28}$$

As r_x defined like in Eq. (S15) is the solution of the quadratic equation $\mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x) = \lambda_x r_x + \Delta\lambda_{\mathcal{V}} r_x^2$, it holds that:

$$\begin{aligned} &\Pr\left(|d_j(x) - d_j(y)| \leq \mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x))\right) \geq \\ &\quad \geq 1 - \gamma \quad \forall y \in B(x, r_x)^S \end{aligned} \tag{S29}$$

We now distinguish between two cases for y: (a) $d_j(y) \leq d_j(x)$ and (b) $d_j(y) \geq d_j(x)$. In case (a) it is trivial: $d_j(y) \leq d_j(x) \leq \mu \cdot \bar{m}_{j,\mathcal{V}}$. Having case (b), Eq. (S29) is equivalent to

$$\begin{aligned} &\Pr\left(d_j(y) - d_j(x) \leq \mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x))\right) \geq 1 - \gamma \\ &\Longleftrightarrow \\ &\Pr\left(d_j(y) \leq \mu \cdot \bar{m}_{j,\mathcal{V}})\right) \geq 1 - \gamma, \end{aligned} \tag{S30}$$

thus Eq. (S16) holds and $B(x, r_x)^S$ is a Lipschitz cap.

Theorem 2 (Convergence via Lipschitz Caps) *Given the tightness factor $\mu > 1$, the set of all sampled points $\mathcal{V}$ and the sample maximum $\bar{m}_{j,\mathcal{V}} = \max_{x \in \mathcal{V}} d_j(x)$. Let the initial ball maximum be defined by $m_j^\star = \max_{x \in \mathcal{B}_0} d_j(x)$. Then:*

$$\forall \gamma \in (0, 1), \exists N \in \mathbb{N} \textit{ s.t. } \Pr(\mu \cdot \bar{m}_{j,\mathcal{V}} \geq m_j^\star) \geq 1 - \gamma \tag{S31}$$

where $N = |\mathcal{V}|$ is the number of sampled points.

Proof. Let $x_j^\star$ be a point such that $d_j(x_j^\star) = m_j^\star$. Given $\gamma \in (0, 1)$ and cap radii r_x as defined in Eq. (S15), we know from the definition of a spherical cap that

$$p_{r_x} = \Pr(B(x, r_x)^S \ni x_j^\star) = \frac{\text{Area}(B(x, r_x)^S)}{\text{Area}\,(\mathcal{B}_0)} \tag{S32}$$

and thus it holds that:

$$\Pr(\exists y \in \mathcal{V}\colon B(y, r_y)^S \ni x_j^\star) = 1 - \prod_{x \in \mathcal{V}} (1 - p_{r_x}) \tag{S33}$$

We derive a lower bound of r_x by using the first sample $x_{j,1}$ and replacing the values in Eq. (S15) as follows:

$$\begin{aligned} &\mu \cdot \bar{m}_{j,\mathcal{V}} - d_j(x) && \text{(S34)} \\ &\quad \geq \mu \cdot \bar{m}_{j,\mathcal{V}} - \bar{m}_{j,\mathcal{V}} = (\mu - 1) \cdot \bar{m}_{j,\mathcal{V}} && \text{(S35)} \\ &\quad \geq (\mu - 1) \cdot d_j(x_{j,1}), && \text{(S36)} \end{aligned}$$

thus a lower bound of all Lipschitz cap radii is given by

$$\begin{aligned} &r_{bound} = \\ &\quad = \frac{-\lambda_x + \sqrt{\lambda_x^2 + 4 \cdot \Delta\lambda_{\mathcal{V}} \cdot (\mu - 1) \cdot d_j(x_{j,1})}}{2 \cdot \Delta\lambda_{\mathcal{V}}} \leq \\ &\quad \leq r_x \quad \forall x \in \mathcal{V} \\ &\Rightarrow \Pr(\exists y \in \mathcal{V}\colon B(y, r_y)^S \ni x_j^\star) \geq \\ &\quad \geq 1 - (1 - p_{r_{bound}})^N \end{aligned} \tag{S37}$$

As in the limit of $N \to \infty$ the probability of Eq. (S37) is 1, it follows that $\forall \gamma \in (0,1)\ \exists N \in \mathbb{N}\colon \Pr(\exists x \in \mathcal{V}\colon B(x,r_x)^S \ni x_j^\star) \geq \sqrt{1-\gamma}$.

Using a set of sampled points $\mathcal{V}$ with cardinality N and using $\hat{\gamma} = 1-\sqrt{1-\gamma}$ as the error rate for the upper bound $\Delta\lambda_x$ of the confidence interval in Eq. (S14). Using the result of Theorem 1, the resulting probability $\forall y \in B(x,r_x)^S$ is:

$$\Pr\left(d_j(y) \leq \mu \cdot \bar{m}_{j,\mathcal{V}}\right) \geq 1-\hat{\gamma} = \sqrt{1-\gamma} \tag{S38}$$

If there is an $x \in \mathcal{V}$ such that $B(x,r_x)^S \ni x_j^\star$, then Eq. (S38) obviously holds also for $x_j^\star$, thus:

$$\Pr(d_j(x^\star) \leq \mu \cdot \bar{m}_{j,\mathcal{V}} | \exists x \in \mathcal{V}\colon B(x,r_x)^S \ni x^\star) \geq \sqrt{1-\gamma}$$

For any sets A, B it holds that $\Pr(A) \geq \Pr(A \cap B) = \Pr(A|B) \cdot \Pr(B)$, and using:

$$A = (\mu \cdot \bar{m}_{j,\mathcal{V}} \geq m_j^\star) \tag{S39}$$

$$B = (\exists x \in \mathcal{V}\colon B(x,r_x)^S \ni x_j^\star) \tag{S40}$$

it follows that $\Pr(\mu \cdot \bar{m}_{j,\mathcal{V}} \geq m_j^\star) \geq \Pr(A|B) \cdot \Pr(B) = 1-\gamma$ and therefore Eq. (S31) holds.

Reachability analysis of nonlinear ODEs

For linear ordinary differential equations (ODEs) there exists a closed-form solution, describing the behavior of the solution-traces over time, for every initial state. For nonlinear ODEs, there is no closed-form solution any more. One is able to calculate the solution for different initial states, but one does not know what happens in between of these already calculated traces.

The main goal in the reachability analysis of nonlinear ODEs, is to over-approximate the reachable states of the ODEs, starting from a set of initial states, such as, an interval, a ball, or an ellipsoid, in a way that one can guarantee that all traces are inside the over-approximation. We call such an over-approximation a *bounding tube*. Let us now define this mathematically:

Definition S1 (Initial value problem (IVP)) *We have a time-invariant ordinary differential equation $\partial_t x = f(x), f : \mathbb{R}^n \to \mathbb{R}^n$, a set of initial values defined by a ball $\mathcal{B}_0 = B(x_0, \delta_x)$ with center $x_0 \in \mathbb{R}^n$ and radius $\delta_0 \in \mathbb{R}$, the initial condition $x(t_0) \in \mathcal{B}_0$ and a sequence of k timesteps $\{t_j : j \in [1, \dots, k] \wedge (t_0 < t_1 < \dots < t_k)\}$. For every t_j, we want to know the solution of*

$$\partial_t x = f(x), \quad x(t_0) \in \mathcal{B}_0 = B(x_0, \delta_x). \tag{S41}$$

The definition can be generalized to time-variant ODEs, as time can be just seen as an additional variable x_{n+1} with $\partial_t x_{n+1} = 1$. Let $\chi(t_j, x_0) = x(t_j)$ be the solution of Eq. (S41) at time t_j, for $x(t_0) = x_0$. In reachability analysis, the goal is to find for every time step t_j an overapproximation $\mathcal{B}_j \supseteq \{\chi(t_j, x) : x \in \mathcal{B}_0\}$, such that the set of these over-approximations build up a bounding tube, containing the reachable states. We define the bounding ball and bounding tube as follows:

Definition S2 (Bounding Ball) *Given an initial ball $\mathcal{B}_0 = B(x_0, \delta_0)$, we call $\mathcal{B}_j = B(\chi(t_j, x_0), \delta_j(\mathcal{B}_0))$ a* bounding ball *at time t_j, if it stochastically bounds the reachable states x at time t_j for all initial points around x_0 having the maximal initial perturbation δ_0.*

As we do not only want to bound the perturbation at one specific time, but on a time series, we define:

Definition S3 (Bounding Tube) *Given an initial ball $\mathcal{B}_0 = B(x_0, \delta_0)$ and bounding balls for $t_0 < \ldots < t_k = T$, we call the series of bounding balls $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$ a* bounding tube.

When using reachability analysis to check for intersections with bad states, it is crucial to compute as tight as possible bounding tubes. Otherwise it would e.g. predict that a car driven by a controller would cause a crash even if the neural network controller is behaving perfectly and never causing a crash. Such wide bounding tubes are thus not useful for actually putting continuous depth-models into operation.