Narrowing the Gap: Towards Analyzable and Realistic Simulators for Safety Analysis of Neural Network Control Systems

Vivian Lin, James Weimer, Insup Lee

Narrowing the Gap: Towards Analyzable and Realistic Simulators for Safety Analysis of Neural Network Control Systems

Vivian Lin, James Weimer, and Insup Lee

PRECISE Center, University of Pennsylvania, Philadelphia PA 19104, USA
{vilin,weimerj,lee}@seas.upenn.edu

Abstract. Although neural network control systems have been used in many applications, they are still challenged by the unreliability of neural networks in the face of unseen data. Thus, it is important to formally verify and assure the safety properties of neural network control systems. A major challenge of this task is a pervading gap between realistic simulators and analyzable simulators of system plants. To narrow this gap, we propose a method to closely approximate a realistic simulator with a less complex simulator. Our approach relies on generative adversarial networks to learn the error statistics between an existing analyzable simulator and an existing realistic simulator. We additionally present a general probabilistic guarantee on the input-output error of an approximation obtained by any method, including our own. We apply our GAN-based approximation technique in a case study of glucose control for type 1 diabetics and evaluate the resulting approximator against data simulated from real patient parameters. We observe that our approach more closely approximates a realistic insulin-glucose simulator than a baseline method.

Keywords: Safety analysis · Neural network control systems · Model approximation · Automatic glucose control.

1 Introduction

Neural networks (NNs) are widely recognized as a powerful tool in modeling and control of complex systems [4, 35, 44]. Despite this, neural network control systems (NNCSs) have significant flaws that continue to plague them during their deployment. For example, NNs have been shown to behave unexpectedly in the real world when data does not match those seen during training [41, 50, 57]. Generally undesirable, this unreliability is an especially relevant concern for safety-critical systems, such as medical devices [18] and autonomous vehicles [4]. Hence, it is important to determine whether users can trust each individual NNCS used in safety-critical systems. One natural step in this process is formally verifying and assuring the safety properties of NNCSs.

Great progress has been made in accomplishing this task. In particular, a common approach is to ensure the safety properties of a *simulated* closed-loop

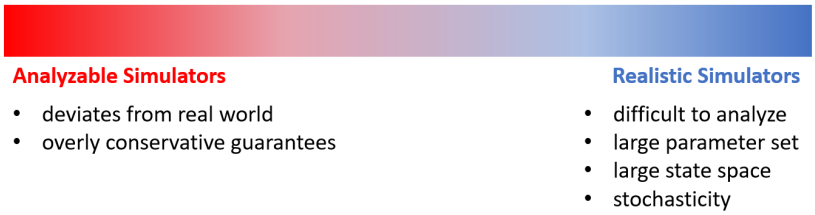


Fig. 1: Spectrum of analyzable and realistic simulators. Existing simulators fall on either end of the spectrum.

system, consisting of the controller and a simulator or model of the plant [12, 16, 32, 33, 56, 63, 64, 70]. However, such solutions face a recurring challenge. Namely, there exists a gap between simulators that are analyzable and simulators that are realistic, as illustrated in Figure 1. Accurate simulators are often so complex, for reasons including a large parameter set [46] or a large state space [29], that the requisite analysis is intractable for existing tools. Conversely, simulators simple enough to enable formal verification and assurance of safety properties are often inaccurate models of the real world. Consequently, in many scenarios it remains difficult to both efficiently and accurately check NNCS safety properties.

To narrow this gap between analyzable and realistic simulators, we propose a method of approximating a realistic complex simulator with an analyzable simulator that maintains the accuracy of the original. For this work, we focus on realistic simulators whose complexity arises from an oppressively large parameter set, which is difficult to efficiently explore, either by sampling-based or more formal methods. In this setting, our broad goal is to find an approximator of the original simulator that has a reduced parameter set. To do so, we use a generative adversarial network (GAN) to learn the error statistics between an existing analyzable simulator and an existing realistic simulator. Our final approximation of the realistic simulator is the combination of the analyzable simulator and our error model. Leveraging the natural structure of GANs, this method allows for a low-dimensional parameter set while maintaining accuracy. Using methods from extreme value theory, we additionally provide a general probabilistic bound on the input-output error between the original simulator and the model approximating the simulator. We note that our technique is applicable to any closed-loop system, but we focus on application to NNCSs due to the elevated difficulty of proving NNCS safety properties.

Our method is particularly useful to NNCSs involved in the medical devices domain. This area of work is commonly subject to difficulties balancing simulator realism and analyzability. As an illustrative example, consider glucose control for type 1 diabetics, in which the goal is to design a controller that automatically selects an appropriate insulin dosage for the patient. The UVa/Padova Type 1 Diabetes Metabolic Simulator (T1DMS) [46] is a comprehensive FDA-approved simulator for the human glucose-insulin dynamics, and hence it is a popular

choice of simulator for checking the safety properties of a glucose controller. However, the T1DMS involves a 32-dimensional parameter set that is too large for existing tools to handle. Furthermore, the parameters corresponding to each individual patient are often partially unknown, leading to discrepancies between even the complex T1DMS and the actual glucose-insulin dynamics of the patient. Such challenges place a significant burden on medical device manufacturers, for whom it is desirable to evaluate a medical device's safety before opting into expensive clinical trials. With these considerations in mind, we select glucose control for type 1 diabetics as a case study of our proposed method, and we develop a GAN-based model with only three parameters to approximate the T1DMS.

In summary, our contributions are 1) a GAN-based approach to learning a reduced parameter set approximation of a simulator, 2) a general probabilistic guarantee on the error between the original simulator and its approximation, and 3) a case study of 2) applied to glucose control for type 1 diabetics with data simulated from real patient parameters.

2 Related Work

There are many approaches to deriving guarantees on systems that involve NNs. The most rigorous area of analysis is formal verification, which can be applied to NNs in both the open-loop and closed-loop settings. In the open-loop setting, the input-output properties of NNs are checked. Methods for open-loop verification include formulating the analysis as an SAT/SMT problem [19, 37], formulating the analysis as an MILP [17, 60, 65], and abstracting the input set as intervals [66], polyhedra [54], zonotopes [24], or star sets [62] to be propagated through the NN. In the closed-loop setting, these methods are extended to perform reachability analysis on a closed-loop dynamical system, composed of an NN controller and some model of a plant. One such approach is to calculate the output range of the NN controller using an open-loop verification method, then propagate this range through some time horizon according to the system dynamics [32, 61]. However, such techniques are often overly conservative, caused by the propagation of the NN output over-approximation error [49]. A variety of alternate methods [16, 33, 56] have been proposed to reduce this over-approximation error in closed-loop NN verification. Despite this extensive literature, formal verification methods struggle to scale with NN size and input size. For example, it has been shown that verifying NNs with ReLU activation functions is NP-complete [37]. As such, other techniques for proving the safety properties of NNCSs have been explored.

One orthogonal approach to formal verification is simulation-based methods. Such techniques use a model of the system and a finite number of initial conditions to produce simulation traces, from which some safety guarantee on the entire set of initial conditions can be derived. In some cases, this safety guarantee is proven using information about the deviation between each simulated trace and some set of unsimulated traces, which can be quantified by sensitiv-

ity analysis [10, 11], discrepancy functions [14, 15], or bisimulation functions [25, 36]. Alternate methods build off of statistical model checking [52, 71] and Monte Carlo methods, sampling the set of initial conditions to compute a probabilistic guarantee on a safety property [8]. Recent work has applied simulation-based techniques to check the input-output properties of NNs [69], perform reachability analysis on NNCSs [64, 70], and perform falsification on NNCSs [12, 63].

Whether through formal verification or simulation-based methods, any approach to checking NNCS safety properties requires a model or simulator of the system plant. Accompanied with this model is an inherent trade-off between how realistic the model is and how analyzable the model is. As a result, researchers have explored methods of reducing model complexity while maintaining accuracy as a means of enabling safety analysis [1, 7, 34, 59]. One area of such literature focuses on reducing the original model to an equivalent abstraction. For example, abstraction has been used to facilitate symbolic model checking [7], which has been applied to safety-critical applications like pacemakers [34]. As another example, since many simple classes of hybrid systems are undecidable for reachability analysis [29], some abstractions of hybrid systems have been explored for verification [1, 59].

Another popular method of balancing the model realism-analyzability tradeoff is using NNs to approximate complex models, which is most similar to our approach. As universal approximators [9, 30], NNs have long been used as functional approximators of system dynamics [22, 40, 55]. More recently, one approach applied this technique to assist in checking the safety properties of closed-loop systems [20]. In a similar vein, another work considered the problem of performing simulation-based reachability analysis on NN-represented nonlinear autoregressive-moving average (NARMA) models that approximate nonlinear dynamics [68]. Finally, one method used NNs to overapproximate models with a probabilistic guarantee, ensuring safety in situations where underestimated values lead to faults [13]. In application, NN approximation for system analysis has shown utility in a number of fields, used to accelerate Monte Carlo simulation methods for analyzing structural failure probabilities [58] and for studying condensed matter systems [53]. This approach has also been used extensively in the verification of glucose controllers, and we refer the reader to Section 5 for a more detailed review of such work. Unlike previous methods using NN approximations to enable safety property assurance and verification, which approximate or abstract a realistic model directly, we construct an approximation by learning the error statistics between a realistic model and an analyzable model.

3 Preliminaries and Problem Statement

We first introduce some notation that will be used throughout the paper. We denote the set of real numbers as $\mathbb{R}$, the set of n-dimensional real vectors as $\mathbb{R}^n$, and the set of strictly positive real numbers as $\mathbb{R}^{++}$. We use $\mathbb{E}$ to refer to the expected value, and $x_i \overset{\text{iid}}{\sim} \mathcal{X}$ to denote that x_i are independently and identically distributed (i.i.d) random variables drawn from the distribution $\mathcal{X}$.

For this work, we abstract simulators and consider them as simple input-output models. Note that this definition of simulators encompasses dynamical models (i.e., the current and next state can be considered the input and output, respectively) that are often used to simulate a real-world environment. We therefore represent a given existing simulator as a model y mapping from input space $\mathbb{X} \subseteq \mathbb{R}^n$ to output space $\mathbb{Y} \subseteq \mathbb{R}^m$, with parameters from some $\mathbb{P} \subseteq \mathbb{R}^p$. We also denote the set of simulator models mapping from $\mathbb{X}$ to $\mathbb{Y}$ with parameter set $\mathbb{Q} \subseteq \mathbb{R}^q$ as $Y_{\mathbb{X},\mathbb{Y},q} = \{y_{\mathbb{Q}} \mid y_{\mathbb{Q}} : \mathbb{X} \to \mathbb{Y},\ \mathbb{Q} \subseteq \mathbb{R}^q\}$.

Our central goal is to approximate the simulator y with some model $\hat{y}_{\mathbb{Q}} \in Y_{\mathbb{X},\mathbb{Y},q}$, with the parameter set of $\hat{y}_{\mathbb{Q}}$ having a dimensionality smaller than the original. The parameters of the new reduced model can then be dynamically chosen using samples of the incoming data. The question of how the parameters should be chosen is out of the scope of this work. Instead, as a first step to addressing the approximation for parameter set reduction problem, we seek a simulator $\hat{y}_{\mathbb{Q}}$ for which there exists some parameter $Q \in \mathbb{Q}$ that minimizes approximation error for each $P \in \mathbb{P}$. More formally, our problem is as follows.

Problem 1 (Approximation for Parameter Set Reduction). Assume that a simulator $y : \mathbb{X} \to \mathbb{Y}$ with parameter set $\mathbb{P} \subseteq \mathbb{R}^p$, a set of models $Y_{\mathbb{X},\mathbb{Y},q}$ with $q < p$, and samples $x_1, x_2, \ldots, x_t \in \mathbb{X}$ are given. Find an approximator of y, denoted $\hat{y}_{\mathbb{Q}}$, that solves

$$\underset{\hat{y}_{\mathbb{Q}} \in Y_{\mathbb{X},\mathbb{Y},q}}{\arg\min} \quad \min_{Q \in \mathbb{Q}} \ \frac{1}{t} \sum_{i=1}^{t} \ell\left(y(x_i), \hat{y}_{\mathbb{Q}}(Q, x_i)\right),$$

where $\ell\left(y(x_i), \hat{y}_{\mathbb{Q}}(Q, x_i)\right)$ is a distance metric that quantifies the distance between $y(x_i)$ and $\hat{y}_{\mathbb{Q}}(Q, x_i)$.

The minimization over $\mathbb{Q}$ in the above optimization problem captures our goal of finding a simulator for which there *exists* a minimizing parameter. Before proceeding with the paper, we provide an illustrative example of Problem 1. Let $\mathbb{X} = \mathbb{Y} = \mathbb{R}$, and let y have parameter set $\mathbb{P} = \{P\}$. Also consider a set of models $Y_{\mathbb{X},\mathbb{Y},q} = \{\hat{y}_{\mathbb{Q}_a},\ \hat{y}_{\mathbb{Q}_b}\}$, where $\mathbb{Q}_a = \{Q_a^1, Q_a^2\} \subseteq \mathbb{R}^{q_a}$ and $\mathbb{Q}_b = \{Q_b^1, Q_b^2\} \subseteq \mathbb{R}^{q_b}$, with $q_a, q_b < p$. Let ℓ be the euclidean distance and the simulator outputs at samples x_1, x_2, x_3, x_4, x_5 be as depicted in Figure 2. In this scenario, there is a parameter selection for $\hat{y}_{\mathbb{Q}_b}$ (i.e., Q_b^1) that achieves the least error out of all four possible approximations of y. Thus, the solution to Problem 1 is $\hat{y}_{\mathbb{Q}_b}$.

4 High-Level Approach

In this section, we describe our high-level approach to Problem 1. We first provide background information on GANs for context, then describe our application of GANs to the problem. We lastly present a probabilistic guarantee on the error of any approximation method, including our own.

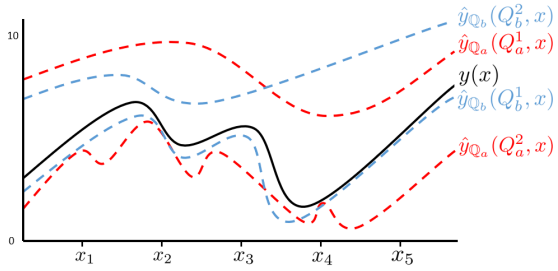


Fig. 2: Example scenario for Problem 1. The solution in this case is $\hat{y}_{\mathbb{Q}b}$.

4.1 Generative Adversarial Networks

The goal of the GAN framework [27] is to train generative neural networks, which generate data from a distribution imitating that of the training data. This method deviates from prior work on generative models by formulating the training process as a two-player game between a generator network G and a discriminator network D. The generator aims to mimic the training distribution as closely as possible, learning a mapping from a prior distribution $p_z(z)$ on a noise input z to the distribution $p_{\text{data}}(x)$ of the training data x. The discriminator, adversary to the generator, aims to determine whether an input is from the true training set or is generated by the generator. More formally, this minimax game is formulated as the following optimization problem:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right].$$

For applications requiring data generation, the discriminator is often used for training only and discarded in deployment. Our solution likewise uses the discriminator for training, but only incorporates the generator in the final simulator. Essential to our approach is the fact that the noise input of the generator can be chosen to have any dimension, although in practice certain choices of dimension are more suitable for training stability. This is discussed further in later sections.

The GAN framework can easily be extended to learn a conditional distribution [48]. In this conditional GAN framework, shown in Figure 3, the generator takes a context (i.e., conditioning) input in addition to a noise parameter and produces a sample from the learned conditional distribution. Although powerful, both GANs and conditional GANs are difficult to train. For example, one common challenge is mode collapse, in which the generator generates only a small subset of the possible outputs [47]. The Wasserstein GAN (WGAN) [3] is a variant of GANs that alleviates these problems. Broadly, the generator of a WGAN mimics the training distribution by minimizing the Earth Mover (or Wasserstein-1) distance [51] between the training and generated distributions. Given these benefits to training stability and preventing mode collapse, we use a conditional Wasserstein GAN in our approach.

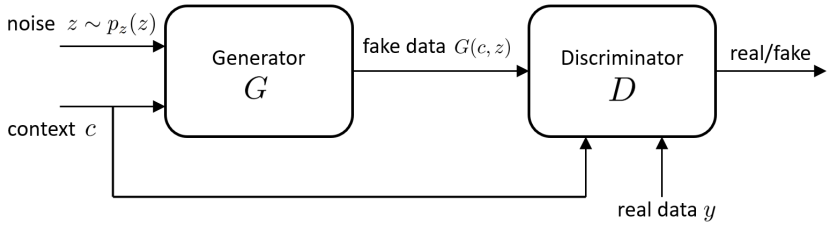
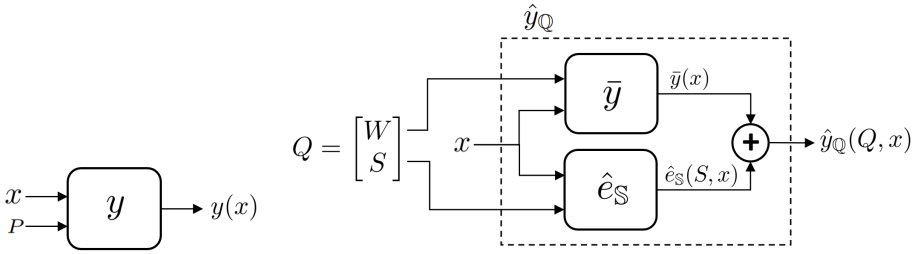


Fig. 3: Block diagram of the conditional GAN structure.



(a) y with parameter $P \in \mathbb{P}$.

(b) $\hat{y}_{\mathbb{Q}}$ with parameter $Q \in \mathbb{Q}$.

Fig. 4: Block diagrams of (a) original simulator with large parameter set $\mathbb{P}$ and (b) approximator of simulator with reduced parameter set $\mathbb{Q}$.

4.2 GAN-Based Parameter Set Reduction

To approximate a given simulator y, we propose to compose two sub-models. Specifically, our approximation $\hat{y}_{\mathbb{Q}}$ of y is given by

$$\hat{y}_{\mathbb{Q}} = \bar{y} + \hat{e}_{\mathbb{S}},$$

where $\bar{y}$ is an existing simple simulator and $\hat{e}_{\mathbb{S}}$ is the generative network of a GAN estimating the error $e = y - \bar{y}$. We denote the parameter set of $\bar{y}$ as $\mathbb{W} \subseteq \mathbb{R}^w$, and we denote the parameter set of $\hat{e}_{\mathbb{S}}$ as $\mathbb{S} \subseteq \mathbb{R}^s$, with $w+s = q < p$. Our construction of $\hat{y}_{\mathbb{Q}}$ is shown as a block-diagram in Figure 4. The core of our approach is training the GAN to learn the error e conditioned on an input x. There are several benefits to this composition approach. First, by exploiting existing simulators, we reduce our task to learning low magnitude error values, which is much simpler than learning the simulator model itself due to increased training stability (smaller NN weights help avoid exploding gradients [26]). Second, as training data we sample input-output pairs from the existing simulators y and $\bar{y}$, allowing us to create a training set of unlimited size.

We now describe each sub-model of our approach in greater detail. To choose $\bar{y}$, we must identify a simple simulator that is meant to accomplish the same task as y and that has a low-dimensional parameter set. That is, $\mathbb{W} \subseteq \mathbb{R}^w$, where $0 \leq w \ll p$. In accordance with the realism-analyzability trade-off discussed earlier,

such a simulator sacrifices accuracy for reduced complexity. To compensate for this less accurate representation of the true plant, we augment $\bar{y}$ with a secondary model $\hat{e}_{\mathbb{S}}$ that approximates the error $e = y - \bar{y}$.

We derive our second model $\hat{e}_{\mathbb{S}}$ from a GAN with parameters $\mathbb{S} \subseteq \mathbb{R}^s$ such that $w + s < p$. Note that GANs naturally lend well to this purpose, as they take a noise parameter with dimensions that can be chosen at the time of design. This sets GANs apart from other architectures that may otherwise be used for time-series data, such as long short-term memory models (LSTMs). We train the generator of this GAN to learn the distribution of the error $e(x) = y(x) - \bar{y}(x)$ conditioned on input x, and we call the resulting generator $\hat{e}_{\mathbb{S}}$. Our final approximation is the sum of the two sub-models, $\hat{y}_{\mathbb{Q}} = \bar{y} + \hat{e}_{\mathbb{S}}$.

We make here a distinction between the predicted error, $e = y - \bar{y}$, and the error of our simulator approximation, $y - \hat{y}_{\mathbb{Q}}$. For the remainder of the paper, we refer to e as the predicted error or error, and we refer to $y - \hat{y}_{\mathbb{Q}}$ as the approximation error.

4.3 Probabilistic Guarantee on Approximation Error

We finally present a probabilistic guarantee on the input-output approximation error between the simulator y and its approximator $\hat{y}_{\mathbb{Q}}$.

Theorem 1 (Probabilistic Approximation Guarantee). *Let $\hat{y} : \mathbb{X} \to \mathbb{Y}$ be any approximation of the model $y : \mathbb{X} \to \mathbb{Y}$, and let $x_1, x_2 \ldots, x_t \in \mathbb{X}$, with $x_i \overset{iid}{\sim} \mathcal{X}$, be samples. Also let $\ell(y(x_i), \hat{y}(x_i))$ be a metric that quantifies the distance between $y(x_i)$ and $\hat{y}(x_i)$. Assume that there exists a sequence $a_t \in \mathbb{R}^{++}$ and a sequence $b_t \in \mathbb{R}$ such that*

$$P_{\mathcal{X}}\left(\frac{\max_{i=1,2,\ldots,t}\{\ell(y(x_i), \hat{y}(x_i))\} - b_t}{a_t} \leq \alpha\right)$$

converges to a non-degenerate distribution function as $t \to \infty$ (i.e., the extreme value theorem assumption). Then, for any $\delta \in (0, 1)$, there exists $\epsilon > 0$ such that, as $t \to \infty$,

$$P_{\mathcal{X}}\left(\max_{i=1,2,\ldots,t}\{\ell(y(x_i), \hat{y}(x_i))\} \leq \epsilon\right) \geq 1 - \delta. \tag{1}$$

Proof. We define the random variable $z_{(t)} = \max_{i=1,2,\ldots,t}\{\ell(y(x_i), \hat{y}(x_i))\}$. By the extreme value theorem, as $t \to \infty$, $z_{(t)} \sim \mathcal{Z}$, where $\mathcal{Z}$ is one of three extreme value distributions, Fréchet, Weibull, or Gumbel. For brevity, we consider here the case that $\mathcal{Z}$ is the Gumbel distribution. The proof follows similarly for the other two cases.

The Gumbel distribution is parameterized by the location μ and scale σ parameters, and it is defined as

$$F_{\mathcal{Z}}(a) = \exp\left\{-\exp\{-(a - \mu)/\sigma\}\right\}.$$

For any $\delta \in (0,1)$, let ϵ be such that $\epsilon \geq -\sigma \ln(-\ln(1-\delta)) + \mu$. Then, by a series of straightforward algebraic manipulations, we have that

$$\begin{aligned} \epsilon \geq -\sigma \ln(-\ln(1-\delta)) + \mu \quad &\iff \quad \exp\{-\exp\{-(\epsilon-\mu)/\sigma\}\} \geq 1-\delta \\ &\iff \quad F_{\mathcal{Z}}(\epsilon) \geq 1-\delta \\ &\iff \quad P_{\mathcal{Z}}(z_{(n)} \leq \epsilon) \geq 1-\delta, \end{aligned}$$

as desired. □

Remark 1. As seen in the proof of Theorem 1, for the Gumbel distribution $\mathcal{Z}$, a choice of ϵ satisfying

$$\epsilon \geq -\sigma \ln(-\ln(1-\delta)) + \mu \tag{2}$$

achieves Equation (1) for any $\delta \in (0,1)$, as $t \to \infty$. In practice, ϵ can also be chosen based on less precise knowledge of the distribution parameters μ and σ. For example, given bounds on the parameters such that $\mu \in (-\infty, \bar{\mu}]$ and $\sigma \in [\underline{\sigma}, \bar{\sigma}]$, it is trivial to show that for any $\delta \in (0,1)$, as $t \to \infty$, Equation (2) holds when

$$\begin{cases} \epsilon \geq -\underline{\sigma} \ln(-\ln(1-\delta)) + \bar{\mu}, & \text{if } \delta \geq 1 - e^{-1}, \\ \epsilon \geq -\bar{\sigma} \ln(-\ln(1-\delta)) + \bar{\mu}, & \text{otherwise.} \end{cases}$$

A similar statement holds for the cases that $\mathcal{Z}$ is the Fréchet or Weibull distribution.

Remark 2. We note that Theorem 1 is agnostic to the approximation technique. Hence, it applies both to the approximation error of our simulator approximation $\hat{y}_{\mathbb{Q}}$ and to any other approximation. Additionally, it relies only on the mild extreme value theorem assumption.

5 Case Study: Glucose Control

We implement our GAN-based approximation approach for parameter set reduction in the safety-critical context of glucose control, including glucose control via an NN controller. In this section, we provide a review of the glucose control problem, a description of our implementation, and finally our experimental results.

5.1 The Glucose Control Problem

For patients with type 1 diabetes, automatic glucose control alleviates the burden of manually calculating and administering insulin dosages. Thus, there has been an extensive effort to develop such controllers, also called artificial pancreases, most commonly using model predictive control [18, 23, 28, 31, 45] or PD/PID control [5, 67]. Figure 5 shows the typical glucose control loop in this setting, in which a controller reads a glucose measurement from the patient and suggests an insulin dosage. The glucose measurements are typically taken from a glucose monitor, and the insulin dosage is supplied via an insulin pump, either in the form of basal (small, continuous dose) or bolus (large, single dose) insulin.

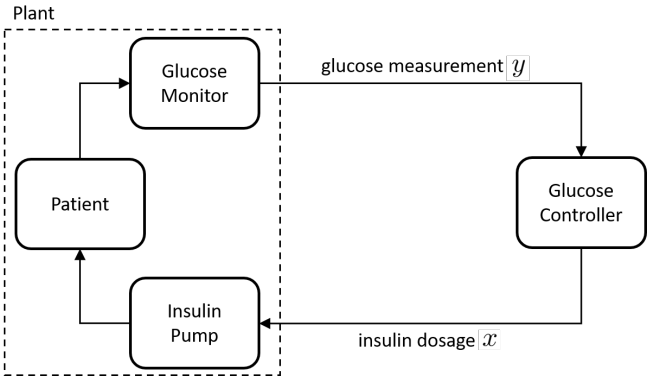


Fig. 5: Typical glucose control loop.

Given the safety-critical nature of this scenario and the high variability between patients, it is important that the safety properties of automatic glucose controllers be verified. In particular, two safety properties of interest are that the controlled glucose levels never rise above a certain threshold (i.e., hyperglycemia) and never fall below a certain threshold (i.e., hypoglycemia). However, analyzing the combined glucose controller and patient to check these properties remains challenging. Such a task is typically performed on the glucose controller and a model or simulator of the patient's glucose-insulin dynamics, but the most accurate models are often so complex that verification is intractable by existing tools. For example, the UVA/Padova Type 1 Diabetes Metabolic Simulator (T1DMS) [46] is both FDA-approved and commonly used in the literature for simulation of insulin-glucose dynamics. The full model relies on 32 parameters to characterize the patient, many of which are often unknown for each patient or difficult to measure. With so many often unknown parameters, it is challenging to verify or assure safety properties using the T1DMS. For example, one work [5] attempted to verify a PD glucose controller against a version of the T1DMS for the surgical setting [6], which has a reduced set of 20 parameters. Using the reachability tool dReach [38] at shallow search depths, the authors could not verify the system for the full range of parameter values.

This difficulty in verifying the safety properties of glucose controllers is a concern to many parties. Patients risk their own health in using an unsafe glucose controller, and regulators serve to protect the patients' interests. Perhaps the most directly affected party are the medical device manufacturers developing tools like glucose controllers. For such manufacturers, clinical trials are an important but expensive step to achieving FDA-approval. Verification of glucose controllers beforehand is one option to safeguard against a failed clinical trial and large scale financial cost. However, especially with the complex biological systems of the human body, verification in the medical domain proves difficult to achieve using accurate models (e.g., T1DMS).

Given the importance and challenges of glucose-insulin modeling for verification, data-driven models of the glucose-insulin system have been explored as alternatives to the T1DMS. Some approaches [2, 18, 43] used NNs trained on patient data and T1DMS-simulated data to predict future blood glucose levels from a history of glucose and insulin readings. These methods either train a single model (e.g., a convolutional NN composed with a recurrent NN [43] or new architecture called the gradually connected NN [2]) or train three small fully-connected NNs to predict the mean, upper quantile, and lower quantile of the future glucose level [18]. Another technique [42] develops a data-driven model of the plant to aid in glucose controller verification. Specifically, an array of linear auto-regressive moving average state-space (ARMAX) models are trained from patient data and then used to predict confidence intervals for a future glucose level based on patient history. These models, which together produce a composite confidence interval, are combined with reachability analysis to predict a reachtube over-approximation of future PID-controlled glucose levels. While the aforementioned data-driven models directly learn the glucose-insulin dynamics using either patient data or data from the T1DMS, our NN learns the error statistics between an existing analyzable simulator and the T1DMS using data simulated from each. From this NN error model and the analyzable simulator, an approximation of the T1DMS with a reduced parameter set can be constructed.

5.2 Implementation

We implement our approximation method for parameter set reduction in the glucose control scenario, constructing a simulator to approximate the T1DMS. In our approach, the T1DMS is the realistic simulator y, with a parameter set of dimension $p = 32$.

For our low-complexity simulator $\bar{y}$, we choose a pre-trained feedforward fully connected NN with 2 hidden layers and 8 neurons per hidden layer designed for glucose control [18]. This simulator has no parameters (i.e., $w = 0$). It takes as input a history H of 37 glucose readings and 37 insulin dosages sampled 5 minutes apart (i.e., a 185-minute history), and it predicts the patient's glucose level 6 samples (i.e., 30 minutes) into the future. More formally, we denote the glucose reading and insulin dosage at time t as g_t and u_t, respectively, and we denote the sampling period as $T = 5$ minutes. Then, given a history $H = \{g_{t-36T}, g_{t-35T}, \ldots, g_t, u_{t-36T}, u_{t-35T}, \ldots, u_t\}$, this simulator predicts g_{t+6T}. The NN is trained against synthetic data collected from an early version of the T1DMS and the clinical trial data of 17 patients. To compensate for the low complexity of the model, the original paper trains two additional NNs to estimate the upper and lower quartiles of g_{t+6T}. We instead propose to learn the error itself of the model compared to the T1DMS.

Finally, for our model $\hat{e}_{\mathbb{S}}$, we use a conditional Wasserstein GAN to learn the distribution of the error e between the T1DMS y and the simple glucose model $\bar{y}$, conditioned on the history H of insulin dosages and glucose measurements. For the discriminator, we use a feedforward fully-connected architecture of 3 hidden layers and 32 neurons per hidden layer. For the generator, we use a feedforward

fully-connected architecture of 3 hidden layers and 64 neurons per hidden layer, with an initial 10% dropout layer during training. In practice, we found that a 3-dimensional noise input (i.e., $s = 3$) sampled from a $Uniform([-10, 10])$ distribution (i.e., $\mathbb{S} = [-10, 10]$) results in the best training results. We train our model on 45,000 (H, e) samples and validate on 15,000 samples. To construct our training and validation sets, we first use the T1DMS to simulate the glucose and insulin traces of 60,000 random synthetic patients (defined by their T1DMS parameters, sampled i.i.d.) for 24 simulation hours. For each patient simulation, we randomly select a sequence of glucose and insulin readings (excluding the first and last two hours of simulation), and parse from it a history H and the corresponding 30-minute look-ahead glucose measurement $y(H)$. We then supply each H to the NN glucose model $\bar{y}$ and calculate the error $e(H) = y(H) - \bar{y}(H)$. The result is a history H and error e pair. Note that samples produced this way are i.i.d., as the T1DMS parameters are sampled i.i.d. and each simulator run is independent of the last. We randomly split the 60,000 (H, e) samples into the training and validation sets.

Our final model approximator $\hat{y}_{\mathbb{Q}}$ is the sum of $\bar{y}$ and $\hat{e}_{\mathbb{S}}$, with only $w + s = 3$ parameters compared to the 32 parameters of the T1DMS. In practice, at each prediction step we supply a history H to both $\bar{y}$ and $\hat{e}_{\mathbb{S}}$, and we supply a three-dimensional noise sample as parameter $S \in \mathbb{S}$ to $\hat{e}_{\mathbb{S}}$. Note that this approach requires some scheme for selecting an appropriate parameter S. Since the correct parameter for each patient could change at each time step, this is a challenging task. One potential solution is to restructure the simulator to predict both the current and future glucose values. The parameter selection can then be made based on the prediction accuracy of the current glucose value, as the true current glucose value is known. The corresponding future glucose value serves as the final prediction. In this paper, we leave further exploration of this parameter selection problem for future work, and we focus on developing an error model $\hat{e}_{\mathbb{S}}$ for which there exists an error-minimizing parameter.

In addition to our model approximator, our probabilistic guarantee for approximation error has utility in the context of glucose control. The guarantee can provide a bound on the likelihood of false negatives when checking for risk of hypoglycemia or hyperglycemia in simulation. The empirical evaluation of our probabilistic approximation guarantee is out of the scope of this paper.

5.3 Results

Note that an evaluation of the error model $\hat{e}_{\mathbb{S}}$ completely characterizes the accuracy of the approximation $\hat{y}_{\mathbb{Q}} = \bar{y} + \hat{e}_{\mathbb{S}}$. Hence, we evaluate our GAN's ability to estimate the conditional distribution of the error e, and we compare it against a baseline approximator. For our baseline approximator, we fit a Laplacian distribution to the training error samples. At test time, for each patient we randomly sample from this fitted distribution to approximate the error.

We compare our GAN and baseline error approximators across a held out dataset simulated using the parameters measured from 22 real patients. Similar to the training and validation sets, to construct this test set we randomly sample

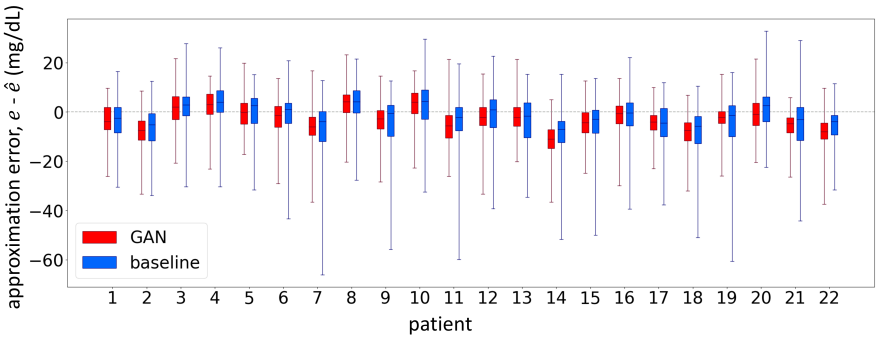
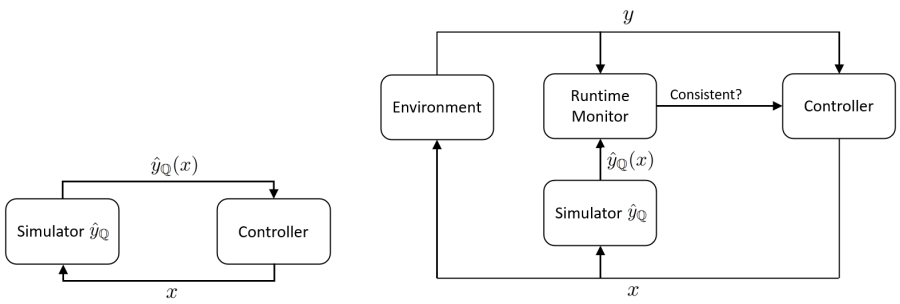


Fig. 6: Comparison of GAN and baseline error approximators over 100 samples for each patient. Boxes represent the interquartile range of approximation error, and whiskers show the full range of approximation error.

a history H from the data of each patient (excluding the first and last two hours of simulation) and record the true error e. Then for each patient, we sample 100 error predictions $\hat{e}$ from the baseline approximator and construct a box and whisker plot of the approximation error $e - \hat{e}$. We also sample 100 $\hat{e}$ predictions (by sampling 100 noise parameter inputs) from the GAN error model $\hat{e}_{\mathbb{S}}$ and construct a box and whisker plot of $e - \hat{e}$. The approximation errors of the baseline approximator and the GAN approximator are represented for each patient in Figure 6 (for scale, an early publication on the T1DMS defined safe blood glucose levels to be between 70 mg/dL and 180 mg/dL [39]). We observe that the GAN approximations for e have a tighter distribution around zero approximation error. That is, the vanilla approximator tends to more grossly overapproximate or underapproximate e than the GAN does. By extension, a model approximator using the baseline error approximator is more likely to yield false positives and false negatives when checking for hypoglycemia and hyperglycemia in simulation. We attribute this result to the fact that the GAN leverages patient-level information (i.e., the 185-minute glucose and insulin history H) to predict the error, unlike the baseline approximator.

6 Discussion and Future Work

In this paper, we proposed a method to learn a simulator approximation with a reduced parameter set, using GANs to learn the error between a highly parameterized simulator and a less parameterized simulator. We applied our method to a case study of glucose control for type 1 diabetics, approximating the 32-parameter T1DMS with a three-parameter simulator. We showed that our GAN-based method is better able to use patient information to approximate a given simulator than a baseline method. We also presented a probabilistic guarantee on the input-output approximation error of any approximation method, including ours. Our proposed method presents a promising first step to building the trust-



(a) Offline verification. (b) Runtime monitoring.

Fig. 7: The two components of an offline verification and runtime monitoring framework.

worthiness of NN-controlled safety-critical systems through efficient but realistic simulators for safety property assurance and verification.

However, questions remain for future work. First, in this paper we discussed a method to design a parameterized model that closely approximates a simulator. How one should choose the best parameter of this model for each scenario is an important next step in our future work. Additionally, our probabilistic guarantee is likely not tight, as it is agnostic to the approximation method. Also, it requires infinite samples. We leave the empirically evaluating the tightness of our bound and deriving a finite-sample version for future work.

Finally, our proposed approximation technique is meant to balance analyzability with realism, but is naturally subject to some approximation error. Hence, we believe our approach is most useful in a framework of offline verification with runtime monitoring, as pictured in Figure 7. In this framework, the desired safety property is verified offline against the simulated closed-loop system, consisting of the simplified simulator $\hat{y}_{\mathbb{Q}}$ and the controller. Online, a runtime monitor checks the consistency of the simulator $\hat{y}_{\mathbb{Q}}$ against real-world data, the result of which aids the controller in making its control decision. When used in such a framework, $\hat{y}_{\mathbb{Q}}$ must be designed to enable both verification and monitoring by existing tools. One potential technique for this task is a variant of knowledge distillation that minimizes the Lipschitz constant of the reduced-order simulator post-hoc [21]. We leave the exploration of this challenge and the implementation of such a framework for future work.

Acknowledgements This work was supported in part by ARO W911NF-20-1-0080 and AFRL and DARPA FA8750-18-C-0090. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force Research Laboratory (AFRL), the Army Research Office (ARO), the Defense Advanced Research Projects Agency (DARPA), or the Department of Defense, or the United States Government.

References

1. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. Proceedings of the IEEE **88**(7), 971–984 (2000)
2. Amar, Y., Shilo, S., Oron, T., Amar, E., Phillip, M., Segal, E.: Clinically accurate prediction of glucose levels in patients with type 1 diabetes. Diabetes technology & therapeutics **22**(8), 562–569 (2020)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International conference on machine learning. pp. 214–223. PMLR (2017)
4. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
5. Chen, S., O'Kelly, M., Weimer, J., Sokolsky, O., Lee, I.: An intraoperative glucose control benchmark for formal verification. IFAC-PapersOnLine **48**(27), 211–217 (2015)
6. Chen, S., Sokolsky, O., Weimer, J., Lee, I.: Data-driven adaptive safety monitoring using virtual subjects in medical cyber-physical systems: a glucose control case study. Journal of Computer Science and Engineering **10**(3), 75 (2016)
7. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM transactions on Programming Languages and Systems (TOPLAS) **16**(5), 1512–1542 (1994)
8. Clarke, E.M., Zuliani, P.: Statistical model checking for cyber-physical systems. In: International symposium on automated technology for verification and analysis. pp. 1–12. Springer (2011)
9. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems **2**(4), 303–314 (1989)
10. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: International Conference on Computer Aided Verification. pp. 167–170. Springer (2010)
11. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: International Workshop on Hybrid Systems: Computation and Control. pp. 174–189. Springer (2007)
12. Dreossi, T., Donzé, A., Seshia, S.A.: Compositional falsification of cyber-physical systems with machine learning components. Journal of Automated Reasoning **63**(4), 1031–1053 (2019)
13. Ducoffe, M., Gerchinovitz, S., Gupta, J.S.: A high probability safety guarantee for shifted neural network surrogates. In: SafeAI@ AAAI. pp. 74–82 (2020)
14. Duggirala, P.S., Mitra, S., Viswanathan, M.: Verification of annotated models from executions. In: 2013 Proceedings of the International Conference on Embedded Software (EMSOFT). pp. 1–10. IEEE (2013)
15. Duggirala, P.S., Mitra, S., Viswanathan, M., Potok, M.: C2e2: A verification tool for stateflow models. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 68–82. Springer (2015)
16. Dutta, S., Chen, X., Sankaranarayanan, S.: Reachability analysis for neural feedback systems using regressive polynomial rule inference. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 157–168 (2019)
17. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks. In: NASA Formal Methods Symposium. pp. 121–138. Springer (2018)

18. Dutta, S., Kushner, T., Sankaranarayanan, S.: Robust data-driven control of artificial pancreas systems using neural networks. In: International Conference on Computational Methods in Systems Biology. pp. 183–202. Springer (2018)
19. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: International Symposium on Automated Technology for Verification and Analysis. pp. 269–286. Springer (2017)
20. Fan, C., Qin, X., Xia, Y., Zutshi, A., Deshmukh, J.: Statistical verification of autonomous systems using surrogate models and conformal inference. arXiv preprint arXiv:2004.00279 (2020)
21. Fan, J., Huang, C., Li, W., Chen, X., Zhu, Q.: Towards verification-aware knowledge distillation for neural-network controlled systems. In: 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 1–8. IEEE (2019)
22. Funahashi, K.i., Nakamura, Y.: Approximation of dynamical systems by continuous time recurrent neural networks. Neural networks **6**(6), 801–806 (1993)
23. Garcia-Tirado, J., Corbett, J.P., Boiroux, D., Jørgensen, J.B., Breton, M.D.: Closed-loop control with unannounced exercise for adults with type 1 diabetes using the ensemble model predictive control. Journal of process control **80**, 202–210 (2019)
24. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 3–18. IEEE (2018)
25. Girard, A., Pappas, G.J.: Verification using simulation. In: International Workshop on Hybrid Systems: Computation and Control. pp. 272–286. Springer (2006)
26. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
27. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
28. Griva, L., Breton, M., Chernavvsky, D., Basualdo, M.: Commissioning procedure for predictive control based on arx models of type 1 diabetes mellitus patients. IFAC-PapersOnLine **50**(1), 11023–11028 (2017)
29. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of computer and system sciences **57**(1), 94–124 (1998)
30. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
31. Hovorka, R., Canonico, V., Chassin, L.J., Haueter, U., Massi-Benedetti, M., Federici, M.O., Pieber, T.R., Schaller, H.C., Schaupp, L., Vering, T., et al.: Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. Physiological measurement **25**(4), 905 (2004)
32. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: Reachnn: Reachability analysis of neural-network controlled systems. ACM Transactions on Embedded Computing Systems (TECS) **18**(5s), 1–22 (2019)
33. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: 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. pp. 169–178 (2019)
34. Jiang, Z., Pajic, M., Alur, R., Mangharam, R.: Closed-loop verification of medical devices with model abstraction and refinement. International Journal on Software Tools for Technology Transfer **16**(2), 191–213 (2014)

35. Julian, K.D., Lopez, J., Brush, J.S., Owen, M.P., Kochenderfer, M.J.: Policy compression for aircraft collision avoidance systems. In: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). pp. 1–10. IEEE (2016)
36. Julius, A.A., Fainekos, G.E., Anand, M., Lee, I., Pappas, G.J.: Robust test generation and coverage for hybrid systems. In: International Workshop on Hybrid Systems: Computation and Control. pp. 329–342. Springer (2007)
37. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: International conference on computer aided verification. pp. 97–117. Springer (2017)
38. Kong, S., Gao, S., Chen, W., Clarke, E.: dreach: δ-reachability analysis for hybrid systems. In: International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems. pp. 200–205. Springer (2015)
39. Kovatchev, B.P., Breton, M., Dalla Man, C., Cobelli, C.: In silico preclinical trials: a proof of concept in closed-loop control of type 1 diabetes (2009)
40. Kumpati, S.N., Kannan, P., et al.: Identification and control of dynamical systems using neural networks. IEEE Transactions on neural networks **1**(1), 4–27 (1990)
41. Kurakin, A., Goodfellow, I., Bengio, S., et al.: Adversarial examples in the physical world (2016)
42. Kushner, T., Bortz, D., Maahs, D.M., Sankaranarayanan, S.: A data-driven approach to artificial pancreas verification and synthesis. In: 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS). pp. 242–252. IEEE (2018)
43. Li, K., Daniels, J., Liu, C., Herrero, P., Georgiou, P.: Convolutional recurrent neural networks for glucose prediction. IEEE journal of biomedical and health informatics **24**(2), 603–613 (2019)
44. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
45. Magni, L., Raimondo, D.M., Bossi, L., Dalla Man, C., De Nicolao, G., Kovatchev, B., Cobelli, C.: Model predictive control of type 1 diabetes: an in silico trial (2007)
46. Man, C.D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., Cobelli, C.: The uva/padova type 1 diabetes simulator: new features. Journal of diabetes science and technology **8**(1), 26–34 (2014)
47. Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.: Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163 (2016)
48. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
49. Neumaier, A.: The wrapping effect, ellipsoid arithmetic, stability and confidence regions. In: Validation numerics, pp. 175–190. Springer (1993)
50. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do imagenet classifiers generalize to imagenet? In: International Conference on Machine Learning. pp. 5389–5400. PMLR (2019)
51. Rubner, Y., Tomasi, C., Guibas, L.J.: A metric for distributions with applications to image databases. In: Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271). pp. 59–66. IEEE (1998)
52. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: International Conference on Computer Aided Verification. pp. 202–215. Springer (2004)
53. Shen, H., Liu, J., Fu, L.: Self-learning monte carlo with deep neural networks. Physical Review B **97**(20), 205140 (2018)

54. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages **3**(POPL), 1–30 (2019)
55. Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.Y., Hjalmarsson, H., Juditsky, A.: Nonlinear black-box modeling in system identification: a unified overview. Automatica **31**(12), 1691–1724 (1995)
56. Sun, X., Khedr, H., Shoukry, Y.: Formal verification of neural network controlled autonomous systems. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 147–156 (2019)
57. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
58. Thaler, D., Elezaj, L., Bamer, F., Markert, B.: Training data selection for machine learning-enhanced monte carlo simulations in structural dynamics. Applied Sciences **12**(2), 581 (2022)
59. Tiwari, A., Khanna, G.: Series of abstractions for hybrid automata. In: International Workshop on Hybrid Systems: Computation and Control. pp. 465–478. Springer (2002)
60. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356 (2017)
61. Tran, H.D., Cai, F., Diego, M.L., Musau, P., Johnson, T.T., Koutsoukos, X.: Safety verification of cyber-physical systems with reinforcement learning control. ACM Transactions on Embedded Computing Systems (TECS) **18**(5s), 1–22 (2019)
62. Tran, H.D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: Nnv: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: International Conference on Computer Aided Verification. pp. 3–17. Springer (2020)
63. Tuncali, C.E., Fainekos, G., Ito, H., Kapinski, J.: Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In: 2018 IEEE Intelligent Vehicles Symposium (IV). pp. 1555–1562. IEEE (2018)
64. Tuncali, C.E., Kapinski, J., Ito, H., Deshmukh, J.V.: Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In: Proceedings of the 55th Annual Design Automation Conference. pp. 1–6 (2018)
65. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. Advances in Neural Information Processing Systems **31** (2018)
66. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1599–1614 (2018)
67. Weinzimer, S.A., Steil, G.M., Swan, K.L., Dziura, J., Kurtz, N., Tamborlane, W.V.: Fully automated closed-loop insulin delivery versus semiautomated hybrid control in pediatric patients with type 1 diabetes using an artificial pancreas. Diabetes care **31**(5), 934–939 (2008)
68. Xiang, W., Lopez, D.M., Musau, P., Johnson, T.T.: Reachable set estimation and verification for neural network models of nonlinear dynamic systems. In: Safe, autonomous and intelligent vehicles, pp. 123–144. Springer (2019)
69. Xiang, W., Tran, H.D., Johnson, T.T.: Output reachable set estimation and verification for multilayer neural networks. IEEE transactions on neural networks and learning systems **29**(11), 5777–5783 (2018)
70. Xiang, W., Tran, H.D., Yang, X., Johnson, T.T.: Reachable set estimation for neural network control systems: A simulation-guided approach. IEEE Transactions on Neural Networks and Learning Systems **32**(5), 1821–1830 (2020)

71. Younes, H.L., Simmons, R.G.: Statistical probabilistic model checking with a focus on time-bounded properties. Information and Computation **204**(9), 1368–1409 (2006)