Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control

HOANG-DUNG TRAN, FEIYANG CAI, MANZANAS LOPEZ DIEGO, PATRICK MUSAU, TAYLOR T. JOHNSON, and XENOFON KOUTSOUKOS, Vanderbilt University

This paper proposes a new forward reachability analysis approach to verify safety of cyber-physical systems (CPS) with reinforcement learning controllers. The foundation of our approach lies on two efficient, exact and over-approximate reachability algorithms for neural network control systems using star sets, which is an efficient representation of polyhedra. Using these algorithms, we determine the initial conditions for which a safety-critical system with a neural network controller is safe by incrementally searching a critical initial condition where the safety of the system cannot be established. Our approach produces tight over-approximation error and it is computationally efficient, which allows the application to practical CPS with learning enable components (LECs). We implement our approach in NNV, a recent verification tool for neural networks and neural network control systems, and evaluate its advantages and applicability by verifying safety of a practical Advanced Emergency Braking System (AEBS) with a reinforcement learning (RL) controller trained using the deep deterministic policy gradient (DDPG) method. The experimental results show that our new reachability algorithms are much less conservative than existing polyhedra-based approaches. We successfully determine the entire region of the initial conditions of the AEBS with the RL controller such that the safety of the system is guaranteed, while a polyhedra-based approach cannot prove the safety properties of the system.

CCS Concepts: • **General and reference** → **Verification**; • **Software and its engineering** → **Formal methods**; • **Theory of computation** → *Timed and hybrid models*; • **Computing methodologies** → *Reinforcement learning*; *Neural networks*; • **Computer systems organization** → *Robotic autonomy*;

Additional Key Words and Phrases: Formal methods, verification, reinforcement learning

ACM Reference format:
Hoang-Dung Tran, Feiyang Cai, Manzanas Lopez Diego, Patrick Musau, Taylor T. Johnson, and Xenofon Koutsoukos. 2019. Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 105 (October 2019), 22 pages.
https://doi.org/10.1145/3358230

This article appears as part of the ESWEEK-TECS special issue and was presented at the International Conference on Embedded Software (EMSOFT) 2019. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers SHF 1736323 and CNS 1739328, the Air Force Office of Scientific Research (AFOSR) through contract number FA9550-18-1-0122, and the Defense Advanced Research Projects Agency (DARPA) through contract number FA8750-18-C-0089. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, or NSF.

Authors' addresses: H.-D. Tran, F. Cai, M. L. Diego, P. Musau, T. T. Johnson, and X. Koutsoukos, Vanderbilt University; emails: trhoangdung@gmail.com, {feiyang.cai, diego.manzanas.lopez, patrick.musau, taylor.johnson, xenofon.koutsoukos}@vanderbilt.edu.

1539-9087/2019/10-ART105 $15.00
https://doi.org/10.1145/3358230

1 INTRODUCTION

Deep neural networks have become a popular choice in practical applications where the control tasks are much more complicated than the traditional control problems. Recently, the power of DNNs has inspired a new generation of intelligent autonomy that makes use of DNNs as learning-based controllers such as autonomous vehicles [7] and air traffic collision avoidance systems [20]. Although utilizing DNNs for intelligent autonomous application is promising, safety verification of autonomy containing neural network components is difficult because DNNs usually have complex characteristics and behavior that are generally unpredictable. Importantly, many pieces of research have proved that well-trained DNNs may not be robust and behave unsafely with a slight change in the input [26]. Recent incidents in autonomous driving (e.g., Tesla and Uber) due to the failures of learning-based components, e.g., misclassifying objects, raises an urgent need for techniques and tools that can formally verify the safety of neural network control systems before utilizing them in safety-critical applications.

Safety verification of neural network control systems (NNCS) is a challenging problem because the behaviors of the systems are difficult to estimate or characterize. To explicitly analyze the safety of NNCS, we need to calculate the exact or overapproximate reachable set containing all possible trajectories of the plant that takes the control set from the neural network controller as inputs. The output set of the plant is feedback to the controller to compute the control set for the next control step. Therefore, if the error in the reachable set computation is large, it quickly becomes larger and larger over time which results in too conservative reachable sets that cannot be used for safety verification. In addition, the scalability and efficiency of the reachable set computation are crucial for safety verification of control systems with DNN controllers. It is required methods that can compute the reachable set of NNCS with large neural network controllers with a reasonable computation time and a small over-approximation error. However, calculating an exact or tight, over-approximate reachable set of a neural network quickly is fundamentally difficult due to the non-linearity of the network. This challenging problem has not addressed well in the existing literature.

In this paper, we propose a new reachability analysis approach for safety verification of CPS with neural network controllers using the concept of star set. We particularly focus on the safety verification of the Advanced Emergency Braking System (AEBS) in an autonomous car to illustrate and evaluate our approach. The AEBS is controlled by a neural network controller which is trained to stop the vehicle appropriately if it discovers an obstacle on the road. To guarantee safety, it is required that the time-to-collision (TTC) of the car, *which is a nonlinear function of the car's velocity, acceleration and the distance between the vehicle to the obstacle*, is always larger than a safe threshold defined by the physical characteristics of the vehicle. Our safety verification approach for AEBS works as follows. First, using CARLA, we perform system identification to obtain a discrete, linear state-space model of the car. The car model is then validated via systematic testing. Second, we train a deep neural network controller to perform the emergency braking action using reinforcement learning. Third, we compose the neural network controller with the state-space model to construct a closed-loop Simulink model of the AEBS which is then validated with CARLA results. Fourth, we perform the reachability analysis of the closed-loop model to obtain the reachable set of the AEBS. Finally, we compute the reachable set of the TTC and use it for safety verification.

We limit our reachability analysis approach to feed-forward neural network controllers with ReLU/Saturation activation functions. Our reachability algorithms can compute both exact and over-approximate reachable sets of the AEBS. Exact reachable set computation is expensive since the number of the reachable sets increases over time steps. In contrast, the over-approximate reachability scheme is much cheaper as it produces a single reachable set at each time step. Importantly, by using star sets, our reachability analysis approach can eliminate or reduce significantly the

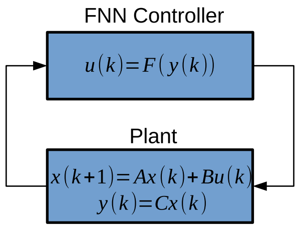

Fig. 1. Neural network control system (NNCS).

over-approximation errors which is the main reason that makes the obtained reachable sets more and more conservative over time as shown in the polyhedron approach [29, 35, 36] (and maybe in some existing methods). Our approach successfully verifies the safety of the AEBS and notably, determine the entire region of the initial conditions of the AEBS where safety is guaranteed. This demonstrates the promising applicability of our approach in verifying safety properties of neural network-based autonomous systems at design time. We note that the polyhedron approach fails to prove the safety property of the system due to its over-approximation errors explode quickly over time.

In summary, the main contributions of this study are as follows.

(1) the provision of star-based reachability schemes designed to efficiently compute the reachable set of an discrete, linear neural network control systems with ReLU activation function,
(2) an end-to-end design and implementation of these schemes in a MATLAB®toolbox called NNV [29] which is publicly available for evaluation and comparison,
(3) and a thorough evaluation on the safety verification of the practical automatic emergency braking system.

2 SYSTEM MODEL AND PROBLEM FORMULATION

2.1 System Model

In this paper, we are interested in safety verification of CPS with neural network controllers as depicted in Figure 1 in which $x(k)$ and $y(k)$ are the state and the output of the plant at the time step k. The controller is a feedforward neural network (FNN) consisting of an input layer, an output layer, and multiple hidden layers. Each layer is comprised of neurons that are connected to the neurons of the preceding layer labeled using weights [18]. The output of the FNN controller, given a specific input vector is determined by three components: the weight matrices $W_{l,l-1}$, representing the weighted connection between neurons of two consecutive layers $l-1$ and l, the bias vectors b_l of each layer, and the activation function f applied at each layer. Formally, the output of a neuron i is defined by:

$$y_i = f(\Sigma_{j=1}^{n}\omega_{ij}x_j + b_i),$$

where x_j is the j^{th} input of the i^{th} neuron, ω_{ij} is the weight from the j^{th} input to the i^{th} neuron, b_i is the bias of the i^{th} neuron. In this paper, we consider FNN controller with the ReLU activation functions defined as $ReLU(x) = max(0, x)$.

2.2 Problem Formulation

Problem 1 (Safety Verification of NNCS). *Given a CPS with an FNN controller F, and a discrete, linear plant P with the initial states $x(0)$ in an initial set X_0, verify whether or not the state of the plant satisfies a safety property in a bounded time steps k_{max}. Formally, we want to verify if*

$\forall x(0) \in X_0 \rightarrow g(x(k)) \models S(g(x(k))), \forall 0 \leq k \leq k_{max}$ in which g is a nonlinear transformation function, S is a linear predicate over the transformed state variables $g(x(k))$ defining the safety requirements of the system.

The core challenges in problem 1 are: 1) given the initial set of states of the plant, how can we efficiently compute the reachable set of the plant over time steps which depends on the control input produced by the FNN controller with nonlinear activation functions, 2) how can we transform the computed reachable set with a nonlinear transformation function to verify the safety property of the system. It is worth to emphasize that ***a small over-approximation error and timing efficiency in reachable set computation are two crucial metrics that determine the applicability of reachability analysis methods in safety verification of practical NNCS***. Therefore, safety verification of NNCS requires computationally efficient methods that can compute the exact or tight over-approximate reachable sets of NNCS in a reasonable time. However, computing the exact or tight over-approximate reachable sets of an FNN is difficult and usually time-consuming. In addition, simple utilization of the control set from the controller to compute the reachable set of the plant may produce a very coarse reachable set which is useless in safety verification. Overcome the challenges in problem 1 is a fundamental step to tackle the following important problem.

Problem 2 (Safety-critical Initial Condition of NNCS). *Given a CPS in problem 1 with the initial states $x(0) \in X_0$, determine the initial condition of the i^{th} state $x^i(0)$ that "may" make the system unsafe while keeping the initial conditions of other states unchanged. We call this initial condition is a "safety-critical initial condition" of the system and assume that the initial conditions of all states are independent.*

Problem 2 is even harder than problem 1 since it is almost impossible to perform backward analysis of CPS with neural network controllers to determine an unsafe initial condition (backward analysis is generally intractable in this case). In the following, we first present our core reachability algorithm for neural network control systems (NNCS). Then, we discuss handling the nonlinear transformation on the computed reachable set for checking the safety of the system, i.e., Problem 1 as well as searching safety-critical initial condition, i.e., Problem 2.

3 REACHABILITY ANALYSIS OF NEURAL NETWORK CONTROL SYSTEMS

The reachability analysis of a NNCS depicted in Figure 1 is done as follows. First, from the initial set of states X_0 of the plant P, the controller F takes the output set of the plant Y_0 as an input to compute the control set $U = F(Y_0)$. Note that Y_0 is an affine mapping of the initial set X_0 with the output matrix C, i.e., $Y_0 = CX_0$. The control set U is then applied to the plant to compute the set of the next state $X_1 = AX_0 + BU$. This routine is performed iteratively to obtain the reachable set of the plant $X_0, X_1, \ldots X_k$, $0 \leq k \leq k_{max}$. To obtain tight reachable sets of the NNCS, we compute the exact control set U given the output set Y. Also, we compute the exact reachable set of state X_k given its initial set X_{k-1} and the corresponding control set U_{k-1}.

3.1 Generalized Star Set

Although computing the exact control set of a FNN controller can be done by the polyhedron approach [29], it is computationally inefficient and not scalable. In addition, the polyhedron-based approach produces a conservative reachable set of the plant because it cannot take advantage of the relationship between U_k and X_k, i.e., $U_k = F(Y_k) = F(CX_k)$. To overcome these challenges, we propose a new reachability analysis approach for NNCS using the concept of star set [4, 5, 30, 31] which is very efficient in affine mapping operation, e.g., $Y_k = CX_k$ and more importantly, it preserves the relationship between U_k and X_k which is crucial to obtain an exact reachable set of the plant. The definition of a star set and its essential properties are given in the following.

Definition 3.1 (Generalized Star Set [4]). A generalized star set (or simply star) Θ is a tuple $\langle c, V, P\rangle$ where $c \in \mathbb{R}^n$ is the center, $V = \{v_1, v_2, \ldots, v_m\}$ is a set of m vectors in $\mathbb{R}^n$ called basis vectors, and $P : \mathbb{R}^m \rightarrow \{\top, \bot\}$ is a predicate. The basis vectors are arranged to form the star's $n \times m$ basis matrix. The set of states represented by the star is given as:

$$[\![\Theta]\!] = \{x \mid x = c + \Sigma_{i=1}^{m}(\alpha_i v_i) \text{ such that } P(\alpha_1, \ldots, \alpha_m) = \top\}. \quad (1)$$

Sometimes we will refer to both the tuple Θ and the set of states $[\![\Theta]\!]$ as Θ. We also restrict the predicate to be a conjunction of linear constraints, $P(\alpha) \triangleq C\alpha \leq d$ where, for p linear constraints, $C \in \mathbb{R}^{p\times m}$, α is the vector of m-variables, i.e., $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, and $d \in \mathbb{R}^{p\times 1}$. A star is an empty set if and only if $P(\alpha)$ is empty.

Proposition 3.2 (Affine Mapping of a Star). *Given a star set $\Theta = \langle c, V, P\rangle$, an affine mapping of the star Θ with the affine mapping matrix W and offset vector b defined by $\bar{\Theta} = \{y \mid y = Wx + b,\ x \in \Theta\}$ is another star with the following characteristics.*

$$\bar{\Theta} = \langle \bar{c}, \bar{V}, \bar{P}\rangle, \bar{c} = Wc + b, \bar{v} = \{Wv_1, Wv_2, \ldots, Wv_m\}, \bar{P} \equiv P.$$

Proposition 3.3 (Star and Half-space Intersection). *The intersection of a star $\Theta \triangleq \langle c, V, P\rangle$ and a half-space $\mathcal{H} \triangleq \{x \mid Hx \leq g\}$ is another star with following characteristics.*

$$\bar{\Theta} = \Theta \cap \mathcal{H} = \langle \bar{c}, \bar{V}, \bar{P}\rangle, \bar{c} = c,\ \bar{V} = V, \bar{P} = P \wedge P',$$
$$P'(\alpha) \triangleq (H \times V_m)\alpha \leq g - H \times c, V_m = [v_1\ v_2 \cdots v_m].$$

We can see that, a star set does not change its predicate over affine mapping operations, and it preserves the center and basis vectors in the intersection with a half-space.

3.2 Exact Reachability Analysis of the Neural Network Controller

The first step in our reachability analysis is to compute the exact control set $U_k = F(CX_k)$ using star-set approach [30]. This computation is done *layer-by-layer* in which the output set of the previous layer is the input set of the next layer. Given a star input set $\bar{\Theta} = \langle \bar{c}, \bar{V}, \bar{P}\rangle$, the reachable set of a layer L can be obtained precisely in two steps. First, an affine map Θ of the input set can be derived quickly with the weight matrix W and bias vector b of the layer, i.e., $\Theta = \langle c = W\bar{c} + b, V = W\bar{V}, P \equiv \bar{P}\rangle$. After calculating the affine map of the input set, the reachable set of the layer $\mathcal{R}_L$ is obtained by applying the ReLU activation function on the affine-mapped set, i.e., $\mathcal{R}_L = ReLU(\Theta)$. Similar to [29], this second step is done by executing a sequence of stepReLU operations $\mathcal{R}_L = ReLU_n(ReLU_{n-1}(\cdots ReLU_1(\Theta)))$. The stepReLU operation on the i^{th} neuron, i.e., $ReLU_i(\cdot)$, works as follows. First, the input star set Θ is decomposed into two subsets $\Theta_1 = \Theta \wedge x_i \geq 0$ and $\Theta_2 = \Theta \wedge x_i < 0$. Note that from Proposition 3.3, Θ_1 and Θ_2 are also stars. Let assume that $\Theta_1 = \langle c, V, P_1\rangle$ and $\Theta_2 = \langle c, V, P_2\rangle$. Since the later set has $x_i < 0$, applying the ReLU activation function on the element x_i of the vector $x = [x_1 \cdots x_i\ x_{i+1} \cdots x_n]^T \in \Theta_2$ will lead to the new vector $x' = [x_1\ x_2 \cdots 0\ x_{i+1} \cdots x_n]^T$. This procedure is equivalent to mapping Θ_2 by the mapping matrix $M = [e_1\ e_2 \cdots e_{i-1}\ 0\ e_{i+1} \cdots e_n]$. Also, applying the ReLU activation function on the element x_i of the vector $x \in \Theta_1$ does not change the set since we have $x_i \geq 0$. Consequently, the result of the stepReLU operation on input set Θ at the i^{th} neuron is a union of two star sets $ReLU_i(\Theta) = \langle c, V, P1\rangle \cup \langle Mc, MV, P2\rangle$. A concrete example of the first stepReLU operation on a layer with two neurons is depicted in Figure 2.

To reduce the computation cost, we can minimize the number of stepReLU operations. This is because if we know that x_i is always larger than zero, then we have $ReLU_i(\Theta) = \Theta$. In other words, we do not need to execute the stepReLU operation on the i^{th} neuron. Therefore, to minimize the number of stepReLU operations and overall computation time, we first determine the ranges of all

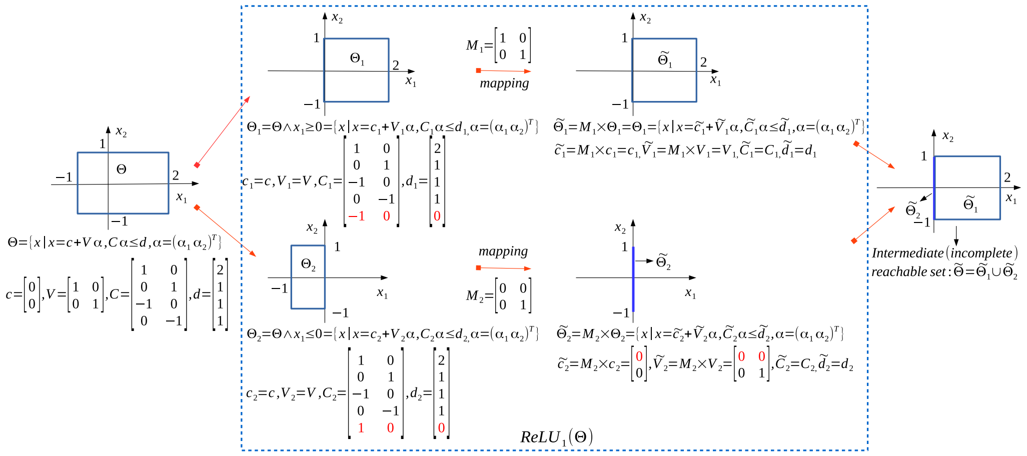

Fig. 2. An example of a stepReLU operation on a layer with two neurons.

states in the input set which can be done efficiently by solving n-linear programming problems. Furthermore, one can see that a star set can be split into two star sets after a stepReLU operation. Therefore, the exact output set of a layer is a union of stars which can be handled independently. Based on this observation, the reachability algorithm of an FNN using star set can be designed efficiently to exploit the power of parallel computing as in the polyhedron-based approach [29]. *We emphasize that the exact reachability of an FNN with ReLU activation function can be extended straightforwardly to deal with saturation activation function.*

Although the star set based method [30] is similar to the polyhedron-based approach [29], it is much more efficient and scalable because star set is very fast in affine mapping which is the most expensive step in the polyhedron-based approach, especially for a high dimensional set. More importantly, the computed output set and the input set of the FNN are defined based on the same set of predicate variables, i.e., $\alpha = [\alpha_1, \ldots, \alpha_m]^T$. This property is crucial in eliminating the over-approximation error in computing the reachable set for the plant as addressed in the following.

3.3 Exact Reachability Analysis of the Discrete Linear Plant

As shown in previous subsection, the exact control set $U_k = F(CX_k)$ is a union of stars, $U_k = \cup_{j=1}^{L}\tilde{\Theta}_j$. Therefore, the exact reachable set of the plant for the next step is also a union of stars, $X_{k+1} = AX_k + BU_k$. Interestingly, the state set $X_k = \langle c, V, P\rangle$ and the control set U_k are defined based on a unique predicate variable vector α and for any star in the control set, its predicate contains all linear constraints of the state set X_k as can be seen in Figure 2. This leads to an important fact that, only a subset of X_k can lead to an individual control set $\tilde{\Theta}_j \in U$ and the predicate of this subset is exactly the predicate of the individual control set. Therefore, the next state set corresponding to the individual control set $\tilde{\Theta}_j = \langle \tilde{c}_j, \tilde{V}_j, \tilde{P}_j\rangle$ is $X_{k+1}^j = \langle Ac + B\tilde{c}_j, AV + B\tilde{V}_j, \tilde{P}_j\rangle$. Consequently, the exact next state set of the plant is $X_{k+1} = \cup_{j=1}^{L} X_{k+1}^j$.

3.4 Reachability Algorithm for NNCS

As shown previously, we can compute the exact reachable set of NNCS depicted in Figure 1 by computing the exact control set and the exact state set of the plant. For a single initial state set, after one time step, it may produce many other state sets. Therefore, the number of state sets increases quickly over time which makes the exact analysis time-consuming even using parallel computing. To handle this state sets explosion, we can obtain a single convex hull of the state sets after every

ALGORITHM 1: Reachability Algorithm for NNCS

```
% $F$: neural network controller
% $A, B, C$: plant's matrices $x_{k+1} = Ax + Bu,\ y_k = Cx_k$
% $I$: initial set of states of the plant
% $k_{max}$: number of steps
% $scheme$: reachability analysis scheme, "exact" or "approx"
% $R$: reachable set
procedure $R$ = Reach($F, A, B, C, I, k_{max}, scheme$)
    $R = cell(1, k_{max} + 1)$
    $R\{1,1\} = I$
    for $k = 1 : k_{max}$ do
        $X_k = R\{1,k\}, M = length(X_k)$
        for $i = 1 : M$ do
            $X_k^i = X_k(i) = \langle c, V, P \rangle$
            $U_k = F(CX_k^i) = \cup_{j=1}^{L} \tilde{\Theta}_j = \cup_{j=1}^{L} \langle \tilde{c}_j, \tilde{V}_j, \tilde{P}_j \rangle$
            $X_{k+1} = []$
            for $j = 1 : L$ do
                $X_{k+1}^j = \langle Ac + B\tilde{c}_j, AV + B\tilde{V}_j, \tilde{P}_j \rangle$
                $X_{k+1} = [X_{k+1}\ X_{k+1}^j]$
        if $scheme == exact$ then $R\{1,k+1\} = X_{k+1}$
        else $R\{1,k+1\} = IntervalHull(X_{k+1})$
```

step and use it for the next step computation. Computing the convex hull for a set of stars is essentially computing the convex hull of a set of convex polyhedrons which is computationally expensive. To overcome this challenge, we instead compute the interval hull of a set of stars for the next step computation which can be done efficiently by solving a set of linear programming optimization problems. The experimental results show that, by using only the interval hull of the star state sets, we still can obtain a tight over-approximation of the exact reachable set for the NNCS and more importantly, the over-approximation error does not explode over time. The reachability algorithm for a NNCS is summarized in Algorithm 1 in which the user can choose to compute the exact or the over-approximate reachable sets of the NNCS.

Lemma 3.4. *The exact scheme in Algorithm 1 produces the exact reachable sets of the NNCS depicted in Figure 1.*

Proof. The proof can be derived inductively based on the exact computation of the reachable set of the plant and the neural network controller in every step. □

3.5 Extension to NNCS with Nonlinear Plants

It is interesting to emphasize that the proposed star-based reachability algorithm can be extended to deal with neural network control systems with nonlinear plants. The core idea of the extension is that we can use existing hybrid systems reachability methods, such as the zonotope-based reachability algorithm in CORA [2] that we chose to use, to compute the reachable set of a nonlinear plant between two time steps t_k and t_{k+1}. This algorithm first further divides the time between t_k and t_{k+1} into N_p smaller time steps, and then performs a sound linearization-based reachable set computation for the plant along with N_p time steps to obtain a reachable set with N_p stars (we note that a zonotope is also a star). We refer readers to [3] for the technical details of this hybrid systems reachability approach. The last star in the union is the initial set of states of the plant for

ALGORITHM 2: Safety Verification for NNCS

```
Input: $R, g, U$: Reachable set of the NNCS, transformation function, unsafe region
Output: $safe = true$ or $safe = uncertain$
1: procedure $safe$ = Verify($R, g, U$)
2:     $k_{max} = length(R)$
3:     for $k = 1 : k_{max}$ do
4:         $X_k = R\{1, k\}$
5:         $\underline{z}_k = min(g(x_k)),\ \bar{z}_k = max(g(x_k)),\ x_k \in X_k$
6:         $\tilde{Z}_k = [\underline{z}_k,\ \bar{z}_k]$
7:         if $\tilde{Z}_k \cap U = \emptyset$ then $safe = true$
8:         else $safe = uncertain$, break
```

the next time interval $[t_{k+1}, t_{k+2}]$. This star is also feedback to the neural network controller. Then, the exact star-based reachability algorithm is invoked to compute the control input U for the next control step.

4 VERIFICATION OF NEURAL NETWORK CONTROL SYSTEMS

4.1 Safety Verification

Although safety properties in CPS are often represented as a linear predicate over the system's states x_k, there are many cases where the safety property is defined as a linear predicate over a variable z_k that is a *nonlinear transformation* of the system's states, i.e., $z_k = g(x_k)$, where g is a nonlinear function. Let $\mathcal{U}(z_k) \triangleq Hz_k \leq h$ be the unsafe region of a NNCS, then safety verification of the NNCS, i.e., Problem 1, is equivalent to checking $Z_k \cap \mathcal{U}(z_k) = \emptyset?\ \forall 0 \leq k \leq k_{max}$, where $Z_k = \{z_k |\ z_k = g(x_k),\ x_k \in X_k\}$ is the transformed reachable set of the system by applying $g(\cdot)$ to it. Since computing the exact transformed reachable set is computationally expensive and may be even infeasible, we compute an over-approximation of the exact transformed reachable set $\tilde{Z}_k$ and use it for safety verification. The system is safe if $\tilde{Z}_k \cap \mathcal{U}(z_k) = \emptyset, \forall 0 \leq k \leq k_{max}$. Particularly, we compute the tightest interval bounding the exact transformed reachable set by solving the following nonlinear optimization problem:

$$\tilde{Z}_k = [\underline{z}_k,\ \bar{z}_k],\ \underline{z}_k = min(g(x_k)),\ \bar{z}_k = max(g(x_k)),\ x_k \in X_k.$$

Safety verification of the NNCS is summarized in Algorithm 2, which solves the above nonlinear optimization problem to obtain the tightest interval of the transformed reachable set and uses it to verify safety of the system at each time step.

4.2 Characterization of Safe Initial Condition

Safety verification of a NNCS can reason about the safety of the system w.r.t a specific initial condition. In some cases, we are interested in the upper bound of a particular state $x^i(0)$ in the initial condition where the safety of the system is still guaranteed. For example, if a car detects an obstacle and applies the brake to stop, it is important to know what is the maximum velocity of the vehicle such that the braking action can guarantee the safety of the car. To search for that maximum velocity, we start from the initial condition that the system is safe, then we increase the upper bound of the speed by some δ, i.e., $x^i(0) = x^i(0) + \delta$, and check the safety of the system with the new initial condition. We continue to increase the upper bound until the safety is uncertain. We can obtain the maximum allowable velocity with the error of $[-\delta, \delta]$.

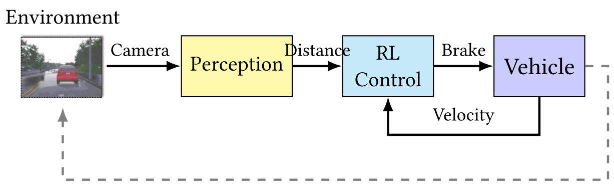

Fig. 3. Emergency braking system architecture.

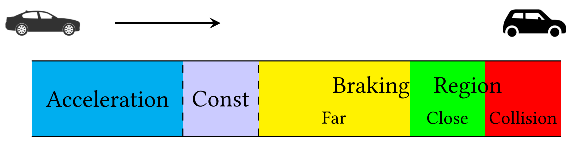

Fig. 4. Illustration of emergency braking system.

5 CASE STUDIES

Our approach is implemented in *NNV* [29, 30], a Matlab toolbox for safety verification of DNNs and learning-enabled CPS. The proposed approach is evaluated on a practical automatic emergency braking system (AEBS) and a adaptive cruise control system (ACC) for an autonomous car. The experiment is done on a computer with following configurations: Intel Core i7-8859H CPU @ 2.6GHz × 4 Processor, 32 GiB Memory, Window 10 Pro OS.[1]

5.1 Advanced Emergency Braking System

The architecture of the AEBS is described in Figure 3 in which the car is equipped with a perception component to detect automatically the obstacle on the road and a reinforcement learning (RL) based controller to control the brake of the car.

5.1.1 Scenario of Interest. In our system, we consider the scenario that the host car automatically detects another static vehicle and applies a brake to decelerate and stop to avoid the potential collision as shown in Figure 4.

The host car starts from rest and accelerates to a random initial velocity v_0, which introduces the uncertainty to the system. Then, the car keeps this velocity v_0 till an obstacle is detected at distance d_0 from the perception module and switches to the reinforcement learning braking controller. The goal of the controller is to stop the car to avoid the collision and also not too far from the obstacle, which means the car should stop within the safety and close region.

5.1.2 Safety Specification. The safety property of the AEBS is defined based on the concept of time-to-collision (TTC) [22, 23]. TTC measures the time it wold take to collide if the vehicle continues traveling based on the current acceleration of $a_k = u_k$ and velocity v_k. Smaller TTC means a higher collision risk. The safety specification of the AEBS can be written by

$$(\mathrm{TTC}_k(d_k, v_k, a_k) > \tau(v_k))\ \mathcal{U}\ (k = k_{max})$$

where $\tau(v_k)$ is the time to stop when applying the full brake for velocity v_k, shown in Figure 5, d_k is the current distance from the car to the obstacle, k_{max} is the maximum number of steps we want to verify the safety of the system, and $\mathcal{U}$ is the until operator. Generally, the safety specification

[1] All results presented in this paper and their corresponding scripts are available online at https://github.com/verivital/nnv/releases/tag/emsoft2019.

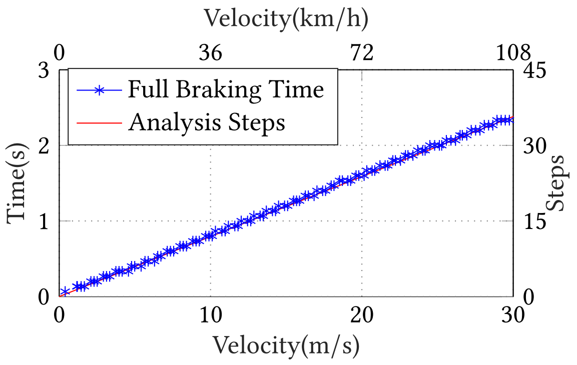

Fig. 5. Obtaining the required number of reachability analysis steps from the full-braking characteristic of the car at different speeds.

means that the car is safe if it still has enough time for a full braking action, i.e., full braking action can successfully stop the car before a collision occurs.

Because of the discontinuity caused by the denominator when velocity or acceleration equals zero, it is more efficient to evaluate the collision risk using the inverse TTC introduced in [6]. The inverse TTC is proportional to the collision risk: the higher it is, the higher the collision risk is. The safety specification using the inverse TTC is given below,

$$(\mathrm{TTC}_k^{-1}(d_k, v_k, a_k) < \tau^{-1}(v_k))\ \mathcal{U}\ (k = k_{max}),$$

where, the inverse TTC is defined by:

$$\mathrm{TTC}_k^{-1}(d_k, v_k, a_k) = \begin{cases} \frac{v_k}{d_k} & \text{for } a_k = 0 \\ \frac{-a_k}{v_k - \sqrt{v_k^2 + 2a_k d_k}} & \text{for } v_k^2 + 2a_k d_k \geq 0 \wedge a_k \neq 0 \\ 0 & \text{for } v_k^2 + 2a_k d_k < 0 \wedge a_k \neq 0. \end{cases}$$

5.1.3 RL-based Controller. We train the RL-based controller for the host car using Deep Deterministic Policy Gradient (DDPG) [24], which is a popular reinforcement learning method that combines the value-based and the policy-based method. There are two parts in this approach including actor and critic. Critic uses the off-policy data to learn the Q-function, which evaluates how good the action a taken is in given state s. The actor can learn the continuous action policy by using the Q-function. In practice, it is difficult to obtain the exact Q-function and policy function. Therefore, two neural networks are introduced to solve this problem, which is critic network $Q(s, a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ. Coming back to our braking system, the reinforcement learning controller consumes the state s, consisting of distance to the leader vehicle d and host car's velocity v, and computes the action – brake T.

For a reinforcement learning system, the reward function should be appropriately designed to achieve the goal. In our case, the task is to stop the car in a safe and close region. Thus, we define the reward function as

$$r = -\alpha \times I \times \mathbf{1}(collision) - [(d_t - B) \times \beta + \lambda] \times \mathbf{1}(v_t = 0 \wedge d_t > B)$$

where d_t and v_t indicates the distance to the leader car and velocity at time step t, α, β and λ are coefficients greater than zero, $\mathbf{1}(\cdot)$ returns a value of 1 if the statement inside is true and 0 otherwise.

The term of the reward function, $-\alpha \times I \times \mathbf{1}(collision)$ penalizes a collision event based on the collision impulse I. The other term $-[(d_t - B) \times \beta + \lambda] \times \mathbf{1}(v_t = 0 \wedge d_t > B)$ penalizes a too early

Table 1. Hyper-parameters for DDPG Algorithm

	Actor	Critic
Optimizer	Adam	Adam
Learning rate	10^{-4}	10^{-3}
Target update rate	0.9	0.9
Reply buffer size	10^5	
Reply batch size	32	
Discount factor	0.99	
Reward function	$\alpha = 0.01, B = 5, \beta = 1.6, \lambda = 20$	

stop based on B, the final distance to the boundary line between close and far region. During the braking process (before the car comes to a stop), there is no penalty or reward. Intuitively, this reward function will guide the car to stop within the close region.

We use CARLA [9] to generate the scenario and to train the reinforcement learning controller. The time step used in the simulation is $\Delta t = 1/15$ s. In the simulations, the vehicle firstly accelerates to a velocity of v_0, and keeps the speed untill it detects an obstacle at a distance d_0. The d_0 and v_0 are the initial states of the braking system. To simulate a more realistic scenario, we introduce some uncertainty to the initial states of the system. The initial velocity of the vehicle is uniformly sampled between 90 km/h and 100 km/h, and the initial distance depends on the range of the perception module, which is approximately 100 m. After initial state, the car switches to the reinforcement learning controller which consists of two neural networks trained with DDPG algorithm with the hyper-parameters in Table 1 is presented below:

- Actor NN architecture[2]:

$$2(\text{State}) \times 50(\text{ReLU}) \times 30(\text{ReLU}) \times 1(\text{SatReLU, Action})$$

- Critic NN architecture[3]:

$$\left.\begin{array}{r} 2(\text{State}) \times 50(\text{ReLU}) \times 30 \\ 1(\text{Action}) \times 30 \end{array}\right\} \times 30(\text{ReLU}) \times 1(\text{Q Value})$$

We trained the reinforcement learning for 1000 episodes, and the neural network converges, showing an attractive performance. Also, one of the experiment trajectories is plotted in Figure 7. At the beginning of involving the reinforcement learning controller, the distance is 97.3 m, and the velocity is 91.98 km/h (= 25.55 m/s). After 128 steps, about 8.53 s, the ego vehicle stops at about 1.88 m far from the obstacle vehicle.

5.1.4 System Identification and Validation. We transfer the braking system from CARLA to MATLAB & Simulink to perform reachability analysis and safety verification for the system. The diagram of the Simulink model of the AEBS is shown in Figure 6. For simulation and verification, only the actor is needed. The plant of the braking system is described by following discrete state-space equation

$$\begin{cases} x_{k+1} & = Ax_k + Bu_k \\ y_k & = Cx_k + Du_k \end{cases}$$

where $x_k = [d_k \; v_k]^T$ is the state vector including the distance d_k and the velocity v_k of the car at step k, u_k is the input, which is the acceleration applied to the plant, y_k is the output, and A, B, C, D

[2]SatReLU is the ReLU function with max value 1.
[3]The empty activation function means no activation is applied.

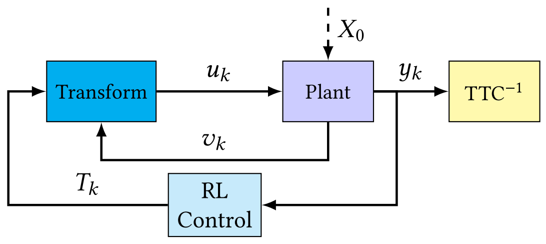

Fig. 6. Emergency braking system simulink diagram.

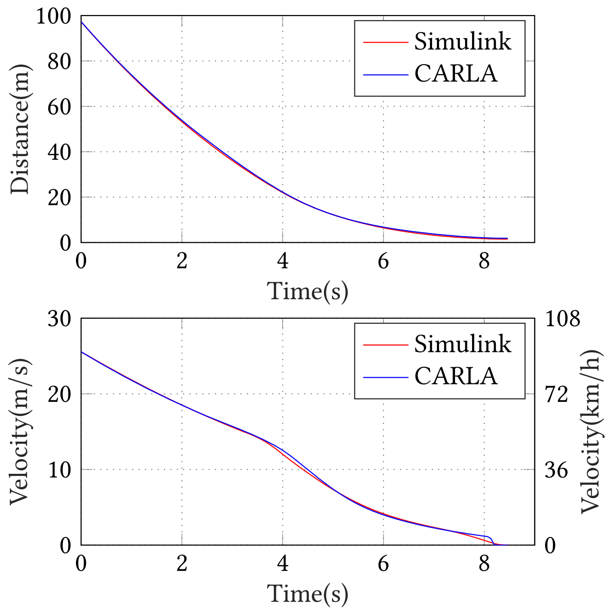

Fig. 7. Validation of the Simulink model of AEBS. The Simulink model captures well the behaviors of the actual AEBS in CARLA.

are the coefficient matrices given below,

$$A = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ \Delta t \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where $\Delta t = 1/15$ is simulation time step.

It is important to emphasize that the input of the plant u_k does not match with the output of the reinforcement learning controller T_k. The u_k is the acceleration applied to the car, but the T_k is the braking force. Thus, a neural network transformation with 80 neurons is trained to bridge this gap between u_k and T_k.

To validate the Simulink model of AEBS, we run experiments in Simulink and CARLA with the same initial states and compare them as shown in Figure 7. From the plot, we can see that the Simulink model captures very well the behaviors of the (actual) AEBS in CARLA.

5.1.5 Safety Verification of the AEBS.

Physical constraints for safety verification. To verify the safety of the AEBS, we need to take into account some essential physical constraints of the system. First, the AEBS system uses

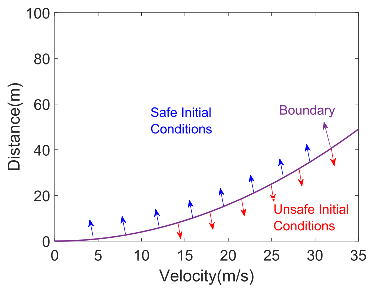

Fig. 8. Safe initial conditions for full braking action.

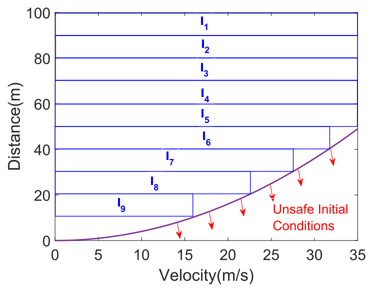

Fig. 9. Set of initial conditions that needs to be verified for the AEBS with the RL controller.

a perception component to detect the obstacle. The operating range of the perception component is from 0 to 100 meters. Therefore, we are going to verify the safety of the AEBS for the distance (between the car to the obstacle) from 10 to 100 meters (we assume that the car is at least 10 meters far away from the obstacle). Secondly, we limit the maximum allowable velocity of the car is 35 m/s, i.e., ≈80 miles per hour which is a usual upper limit of the speed on highways.

Thirdly, we need to know what is a reasonable constraint between initial conditions of the car's velocity and its distance to the obstacle such that if we apply a full braking action, the car is safe. This information is important that we should know before verifying the safety of AEBS because there are cases when even if we apply the full braking action, the collision still occurs. For example, the car is too close to the obstacle and is travelling at a high speed. From Figure 5, we approximate an analytical formula for the full braking time that is $\tau(v) \approx v/12.5$. When the full braking action occurs, the car goes a distance $d_s = 0.5a\tau^2 + v\tau$ before stopping, where a is the average acceleration of the car which is equal to $a = \Delta v/\Delta t = (0 - v)/\tau$. Therefore, the average travel distance of the car after applying a full brake is: $d_s = 0.5v\tau = 0.5v^2/12.5 = v^2/25$. To guarantee the safety, the initial distance of the car d_0 should be larger than this travel distance, i.e., $d_0 > d_s$. Combining the above limitation on the distance $d_{max} = 100$ m and the maximum allowable velocity $v_{max} = 35$ (m/s), a ***safe initial condition region for full braking action*** is depicted in Figure 8. By partitioning the safe initial condition region of the full braking action, we can derive ***the reasonable initial conditions that need to be verified for the safety of the AEBS with the RL controller*** as shown in Figure 9. This is because, under the safety aspect, the RL controller cannot overcome the full-braking action.

Finally, we need to find out what the minimum number of steps that we should at least give a guarantee about the safety of the system is. We should prove the safety of the system at least $\tau(v)$ seconds in the future where $\tau(v)$ is the full braking time w.r.t the velocity v. Therefore, ***the minimum number of steps that needs to prove the safety*** is: $min(k_{max}) = \tau(v)/\Delta t$. For example, if

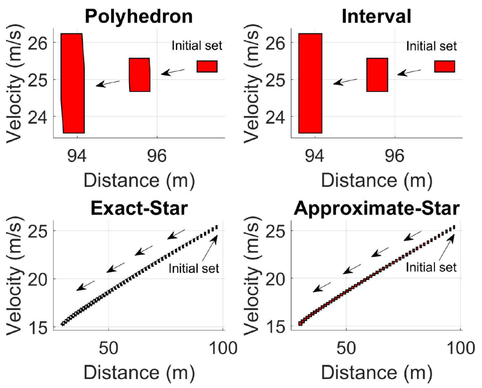

Fig. 10. Reachable sets of the AEBS computed by the polyhedron and the interval approaches become too conservative are quickly after only 2 time steps while the proposed star methods obtains the exact or tight over-approximate reachable sets for many time steps. The initial conditions are $d_0 \in [97,\ 97.5]$, $v_0 \in [25.2,\ 25.5]$.

$v = 25$ m/s, we should at least prove the safety of the system until $k = k_{max} = 2/(1/15) = 30$ time steps.

Challenges and drawbacks of the polyhedron [29, 36] and interval [11] approaches. A main challenge in safety verification of AEBS is how to compute a tight reachable set of the AEBS model depicted in Figure 6. One can see that the control set $U = \{u_t\}$ applied to the plant is derived from the transformation component that takes the output set $T = \{T_t\}$ from the RL controller and the velocity $V = \{v_t\}$ as the input set. Therefore, to compute the control set U, we need to compute the output set T of the RL controller and then combine with the velocity set $V = \{v_t\}$ of the plant to form the input set for the transformation neural network. The problem is how to efficiently combine these sets to form the exact input set for the transformation neural network. This problem is unsolvable if we use the polyhedron-based [29, 36] or the interval [11] methods since the relationship between the output set T of the RL controller and the velocity set V of the plant cannot be preserved in the computation. This leads to a coarse combination which returns a coarse input set for the transformation neural network. Consequently, the over-approximation error is exploded quickly after only 2 time steps as shown in Figure 10 which makes the obtained reachable sets become too conservative and cannot be used for safety verification.

Minimizing overapproximation while maintaining scalability with star sets. As shown in Figure 10, our star-based approach is an efficient technique to overcome the main challenges discussed above. We compute the reachable set for the AEBS system in 50 steps. One can see that our star-based approach eliminates (in the exact method) or reduces significantly (in the over-approximation method) the over-approximation errors caused by the polyhedron-based and the interval approaches. The reachable sets computed from the star-based approach are tight and useful for safety verification of the AEBS.

Error analysis. Since we can compute the exact reachable sets of the system, we can analyze the overapproximation errors of different approaches. These overapproximation errors are measured using the following metrics $OverApprox - Error = max(|lb - lb_{exact}|, |ub - ub_{exact}|)$ which measures that largest distance between the lower- and upper- bounds of an output computed from the overapproximation methods and the exact lower- and upper-bounds computed by the exact star method. The overapproximation errors of the polyhedron, interval, and the proposed over-approximate star methods are depicted in Figure 11. One can see that the overapproximation errors of the polyhedron and the interval methods are increased quickly after only two steps while

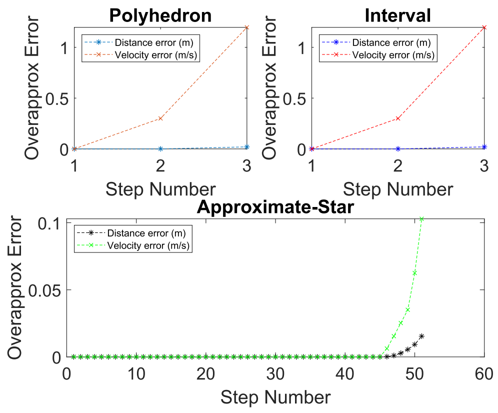

Fig. 11. The Over-approximation errors of the polyhedron and the interval approaches are exploded quickly after only 2 time steps. These over-approximation errors are reduced siginificantly by the proposed star method.

Table 2. Reachability Analysis Times (Measured in Seconds) of the Exact and Over-approximate Star Methods in which N is the Number of Time Steps

Method	**N = 5**	**N = 10**	**N = 20**	**N = 30**	**N = 40**	**N = 50**
Exact star	12.47	32.24	162.95	400.13	532.1	831
Over-approximate star	10.07	21.09	42.86	63.98	83.3	104.44
Time improvement	**1.24x**	**1.53x**	**3.8x**	**6.25x**	**6.39x**	**7.96x**

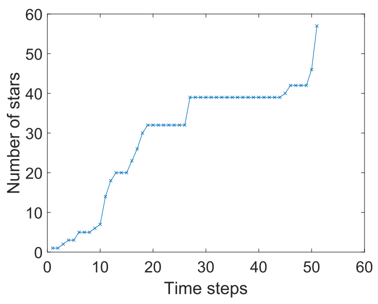

Fig. 12. Number of stars in the reachable sets of the AEBS grows over time with the exact star-based method.

these errors are almost zeros for the first 45 steps and very small in the last five steps in the over-approximate star method.

Timing performance. The reachability analysis times of two proposed methods are presented in Table 2. The Table shows that the over-approximation method is faster than the exact method while still produces tight reachable sets for the system. From the figures, one can see that the reachable sets computed by the two methods are almost the same. The time improvement of using the over-approximation method increases as the number of time steps grows. The reason that makes the over-approximation method faster is, it produces only a single reachable set at every time step while in the exact method, the number of reachable sets may grow over time as depicted in Figure 12.

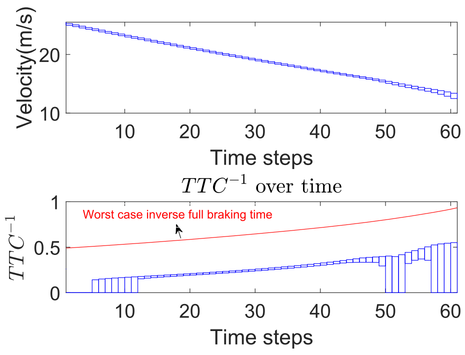

Fig. 13. The inverse TTC over time is smaller than the worst case inverse full braking time $\tau^{-1}(max(v))$. The AEBS is safe (for 60 time steps) with the initial conditions $d_0 \in [97,\ 97.5]$, $v_0 \in [25.2,\ 25.5]$.

Table 3. Safe Initial Conditions for the AEBS with RL Controller in which d is the Distance Range and v_{max} is the Maximum Allowable Velocity Such that the System is Still Safe

$d(m)$	[0, 10]	[10, 20]	[20, 30]	[30, 40]	[40, 50]
$v_{max}(m/s)$	—	4	7	8	10
$d(m)$	[50, 60]	[60, 70]	[70, 80]	[80, 90]	[90, 100]
$v_{max}(m/s)$	15	19	21	24	26

Checking safety using the computed reachable sets. To verify the safety of the AEBS, we consider the worst case, i.e., we want to verify if the following constraint is satisfied in a bounded time, $max(TTC^{-1}(d, a, v)) < \tau^{-1}(max(v))$. To do that, we estimate the ranges of TTC^{-1} in 60 time steps (two times larger than the minimum requirement $k_{max} = 30$) using the ranges of the distance, velocity, and acceleration of the car from the computed reachable sets and check if it satisfies the requirement or not. The result is illustrated in Figure 13 which shows that the inverse TTC is smaller than the worst case inverse full braking time $\tau^{-1}(max(v))$. Therefore, the AEBS is safe for 60 time steps in the future.

5.1.6 Safe Initial Conditions of the AEBS. From the physical constraints of the car, we have derived the set of initial conditions that need to be verified for the AEBS with RL controller as depicted in Figure 9. It is important to determine in these initial conditions, which regions are safe for the AEBS with RL controller and which ones are risks. We perform our safety verification methods on each partition $I_i, i = 1, 2, \ldots, 9$ of the initial conditions to find the safe regions. We perform the search as follows. We partition the distance range [10, 100] into 9 smaller ranges with the same width of 10, i.e., $d^i = [10i, 10(i+1)], 1 \leq i \leq 9$. For the i^{th} individual distance range, we search for the maximum velocity v^i_{max} such that the RL controller can guarantee the safety of the system in $k_{max} = 50$ time steps for the initial condition of $[d^i, v^i_{max}]$. The results of v_{max} are presented in Table 3. From the information of v_{max}, we visualize the safe region of the initial conditions for the AEBS as depicted in Figure 14.

5.2 Adaptive Cruise Control System

The extension of the proposed star-based reachability algorithm is evaluated on the safety verification of neural network-based adaptive cruise control systems (ACC). The ACC system consists

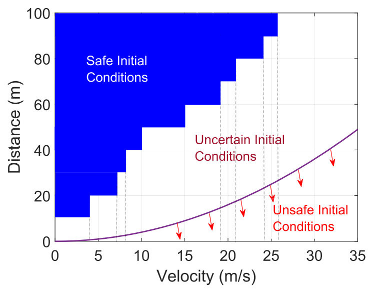

Fig. 14. Safe region of the initial conditions of the AEBS with the RL controller.

of two cars in which the ego car equipped with adaptive cruise control has a radar sensor to measures the distance to the lead car in the same lane, D_{rel}, as well as the relative velocity of the lead car, V_{rel}. In speed control mode, the ego car travels at a driver-set speed $V_{set} = 30$ while in spacing control mode, the ego car maintains a safe distance from the lead car, D_{safe}. Neural network adaptive cruise controllers with different sizes are trained to replace the existing MPC controller. The control period is selected as 0.1 seconds. The car's dynamics are as follows.

$$\dot{x}_{lead}(t) = v_{lead}(t),\ \dot{v}_{lead}(t) = \gamma_{lead},$$
$$\dot{\gamma}_{lead}(t) = -2\gamma_{lead}(t) + 2a_{lead} - \mu v_{lead}^2(t),$$
$$\dot{x}_{ego}(t) = v_{ego}(t),\ \dot{v}_{ego}(t) = \gamma_{ego},$$
$$\dot{\gamma}_{ego}(t) = -2\gamma_{ego}(t) + 2a_{ego} - \mu v_{ego}^2(t),$$

where $x_{lead}(x_{ego})$, $v_{lead}(v_{ego})$ and $\gamma_{lead}(\gamma_{ego})$ are the position, velocity and acceleration of the lead (ego) car respectively, $a_{lead}(a_{ego})$ is the acceleration control input applied to the lead (ego) car, and $\mu = 0.0001$ is the friction parameter.

Safety-related scenario. The safety verification scenario of interest is that when the ego is in the speed control mode and the two cars are running with a safe distance between them, the lead car driver suddenly de-accelerate with $a_{lead} = -2$ to reduce the speed. We expect that the neural network controllers will also de-accelerate the ego car to remain a safe distance between two cars. Formally, the safety specification of the system is $D_{rel} = x_{lead} - x_{ego} \geq D_{safe} = D_{default} + T_{gap} \times v_{ego}$, where $T_{gap} = 1.4$ seconds and $D_{default} = 10$. We want to check if there is a collision in the next 5 seconds after the lead car de-accelerate. The initial conditions of the system are: $x_{lead}(0) \in [90, 110]$, $v_{lead}(0) \in [32, 32.2]$, $\gamma_{lead}(0) = \gamma_{ego}(0) = 0$, $v_{ego}(0) \in [30, 30.2]$, $x_{ego} \in [10, 11]$.

Verification results. The verification results are presented in Table 4 which shows that the second controller is the safest controller since it guarantees the safety of the ACC system for the whole range of the lead car's initial position. The safety of the system can be observed intuitively via Figure 15 which shows that the relative distance is larger than the safe distance. Interestingly, the controllers with a large number of neurons, e.g., the third and the fourth controllers, are not necessarily the good candidates for keeping the system safe. In many cases, the verification results for these controllers are uncertain which imply that these controllers may or may not control the system safely. In these cases, the relative distance reachable set intersects with the safe distance reachable set. However, we do not know this intersection is due to the over-approximation error of the related reachable sets or the relative distance is actually smaller than the required safe distance. Since the safety of the system may be violated in these cases, we further randomly generate simulation traces of the system to find counter example inputs that make the system unsafe. Interestingly, we cannot find counter examples for the system whenever $x_{lead} \in [70, 110]$. However, when $x_{lead} \in [65, 70]$, we can find counter examples of the system for all controllers. Note that in

Table 4. Verification Results for the ACC System in which VT is the Verification Time and Controller $k \times n$ Means the Controller has k Hidden Layer and n Neuron Per Layer

$x_{lead}(0)$	**Controller 1 (3x20)**		**Controller 2 (5x20)**		**Controller 3 (7x20)**		**Controller 4(10x20)**	
	Result	VT (sec)	Result	VT (sec)	Result	VT (sec)	Result	VT (sec)
[108, 110]	safe	211.84	safe	292.67	safe	398.41	safe	1762
[106, 108]	safe	210.16	safe	288.83	safe	393.35	safe	2270.3
[104, 106]	safe	211.54	safe	302.31	safe	412.81	safe	2674.5
[102, 104]	safe	215.21	safe	292.94	safe	446.47	safe	2863.8
[100, 102]	safe	222.87	safe	294.81	safe	440.94	safe	2606
[98, 100]	safe	233.02	safe	302.74	safe	491.29	uncertain	2855
[96, 98]	safe	237.12	safe	289.87	safe	515.43	uncertain	3249.9
[94, 96]	safe	238.46	safe	301.99	uncertain	571.75	uncertain	3851.5
[92, 94]	safe	259.46	safe	325.51	uncertain	598.22	uncertain	3220.2
[90, 92]	uncertain	265.29	safe	359.92	uncertain	558.65	uncertain	2336.9

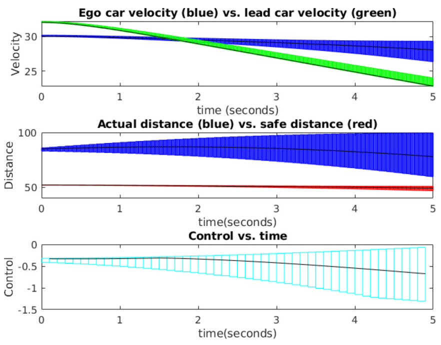

Fig. 15. The reachable sets of the ACC system with the second controller (5x20) and $x_{lead}(0) \in [94, 96]$.

this case, even the controllers de-accelerates the ego car. It still can not guarantee the safety for the system.

Timing performance. As depicted in Table 4, the verification time depends on the size of the controller. A controller with a large number of neurons causes a substantial verification time. Our approach can prove the safety of the NNACC system with the fourth controller having totally 200 neurons in some cases with reasonable verification times (less than 1 hour). A brief comparison with recent approaches [19, 27, 28] is given as follows. Verisig takes averagely 1690 seconds (on their personal computer) to verify a single safety property (corresponding to a single input set) in 30 time steps of the quadrotor system with 12 state variables and a neural network controller having 40 neurons while our approach spends averagely 275.68 seconds to verify a single safety property of the ACC system with 6 state variables and a neural network controller having 100 neurons in 50 time steps. The SMC-based approach can falsify safety property of neural network control system with fairly large number of neurons in the controller (22 to 182 neurons). However, in the case that there is no counter example exist, the SMC-based approach usually reaches timeout (= 1 hour). Its experimental results show that only three controllers (in 17 controllers) with 22, 32 and 82 neurons are successfully verified. An important factor making our approach potentially faster and more scalable than the Verisig and SMC-based approaches is, our approach can efficiently compute the exact reachable set of DNNs on multi-core platforms. Therefore, our

Table 5. Verification Approaches for NNCS

Approaches	**Plant Dynamics**	**Discrete/Continuous**	**Activation Function**	**Size of Controller**
Polyhedron-based [36]	Linear	Discrete	ReLU	$\leq$ 100 neurons
Verisig [19]	Linear, Nonlinear	Discrete, Continuous	Sigmoid, Tanh	$\leq$ 50 neurons
SMC-based [28]	Linear	Discrete	ReLU	$\leq$ 200 neurons
Sherlock [27]	Linear, Nonlinear	Discrete, Continuous	ReLU	$\leq$ 500 neurons
Star-based	Linear, Nonlinear	Discrete, Continuous	ReLU	$\leq$ 200 neurons

verification time for neural network control system can be reduced significantly by exploiting the power of parallel computing. The new abstraction-based approach proposed recently in [27] is promisingly the fastest and the most scalable approach for safety verification of NNCS since it can compute the reachable set of NNCS with neural network controller of 500 neurons in 50 time steps with just 1081 seconds.

6 RELATED WORK

Verification, testing and falsification of CPS with learning-enabled components have become an emerging research topic recently. Toward verification for CPS with learning enable components, several methods have been proposed recently to verify the safety of feedback neural network control systems [11, 19, 27, 28, 34, 36]. The early polyhedron-based approach has been extended for safety verification of neural network controlled systems in [36] in which the plant is assumed to be *linear and discrete*. Recently, Verisig [19] proposes an approach that transforms a neural network controller with *sigmoid* activation function to an equivalent nonlinear hybrid system which is then combined with the plant dynamics before utilizing Flow* [8] to verify its safety properties. Another approach using satisfiability modulo convex (SMC) solver for formal verification of NNCS has been proposed in [28]. In this context, the closed-loop control system was encoded as monotone SMC formulas which were formally verified by SMC decision procedures. Impressively, this method is sound and complete with noticing that the plant dynamics is *linear and discrete*. Last but not least, a new abstraction method [27] has been proposed for NNCS verification in which an "local" Taylor model over-approximation of neural network controller was obtained and integrated into Flow*, a flowpipe constructor using Taylor model, to compute a tight over-approximation reachable set of NNCS. This method is impressively fast and scalable for NNCS verification and more importantly, it can reduce over-approximation errors significantly in reachable set computation process and can deal with relative large input sets. A summary of recent verification methods is given in Table 5. In the testing context, a simulation-based test generation framework for autonomous vehicles with machine learning components has been proposed in [33] to enhance the reliability of autonomous driving systems. In the falsification context, a compositional falsification framework for CPS with machine learning components has been proposed in [10]. In this framework, a temporal logic falsifier cooperates efficiently with a machine learning analyzer to find falsifying executions of the system. The effectiveness of the proposed framework was shown via Automatic Emergency Braking System (AEBS). As a complement approach to verification of CPS with learning-enabled components, in this paper, we focus on safety verification of NNCS with ReLU activation function. We mainly focus on the exact and over-approximate analysis for such a system which aims at eliminating or significantly reducing the over-approximation error in the reachable set computation. We also study an extension of our approach in combination with the zonotope-based reachability algorithms [2, 16] to deal with NNCS with nonlinear plant.

Safe reinforcement learning [12] (SRL) is an essential research topic in safety-critical applications such as medical robotics and self-driving cars. The fundamental challenge in SRL is that the agent not only needs to learn policies that maximize the long-term return based on the reward signal but also ensure the safety constraints in learning or deploying processes. There are two main approaches to safe reinforcement learning. The first one [14, 15] modifies the optimality criterion with the integration of the safety constraints while the second [1, 13, 17, 21, 25] is based on the modification of the exploration process with the incorporation of the prior knowledge or guidance of probabilistic risk metric. Our approach benefits the current state-of-art of safe reinforcement learning methods since it can formally verify the *safety of the learned safe control policies in the interaction with physical world*, i.e., physical dynamics of the agent and the environment. Importantly, many safe reinforcement learning methods usually quantify the risk by some probabilistic distribution and use it for learning and deploying processes. Therefore, the expected safety of the system is usually quantified in a probabilistic manner. In contrast, our verification approach gives a specific answer, i.e., safe, unsafe, or uncertain, about the safety of the system.

7 CONCLUSION

We have proposed two efficient, exact and over-approximate reachability schemes and an optimization-based approach for safety verification of CPS with RL controller where the safety specification is defined based on a nonlinear transformation of the system states. From thorough experiments on the practical AEBS, we have shown that our method is computationally cheaper and less conservative than the existing polyhedron approach. More important, it is applicable to real-world applications. We have also studied an extension of our approach for verification of NNCS with nonlinear plant and evaluate its potential via the NNACC system. Our future work is extending the proposed methods for nonlinear NNCS with other types of nonlinear activation functions such as Tanh or Sigmoid. We are also investigating the runtime verification for CPS with learning-enabled components leveraging the recent promising approach proposed in [32] in combination with the neural network reachability analysis methods.

REFERENCES

[1] Anayo K. Akametalu, Jaime F. Fisac, Jeremy H. Gillula, Shahab Kaynama, Melanie N. Zeilinger, and Claire J. Tomlin. 2014. Reachability-based safe learning with Gaussian processes. In *53rd IEEE Conference on Decision and Control*. IEEE, 1424–1431.

[2] Matthias Althoff. 2015. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*.

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

[4] Stanley Bak and Parasara Sridhar Duggirala. 2017. Simulation-equivalent reachability of large linear systems with inputs. In *International Conference on Computer Aided Verification*. Springer, 401–420.

[5] Stanley Bak, Hoang-Dung Tran, and Taylor T. Johnson. 2019. Numerical verification of affine systems with up to a billion dimensions. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 23–32.

[6] Valentina E. Balas and Marius M. Balas. 2006. Driver assisting by inverse time to collision. In *2006 World Automation Congress*. IEEE, 1–6.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[8] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*. Springer, 258–263.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. *arXiv preprint arXiv:1711.03938* (2017).

[10] Tommaso Dreossi, Alexandre Donzé, and Sanjit A. Seshia. 2017. Compositional falsification of cyber-physical systems with machine learning components. In *NASA Formal Methods Symposium*. Springer, 357–372.

[11] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. 2018. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine* 51, 16 (2018), 151–156.

[12] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[13] Clement Gehring and Doina Precup. 2013. Smart exploration in reinforcement learning using absolute temporal difference errors. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 1037–1044.

[14] Peter Geibel and Fritz Wysotzki. 2005. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research* 24 (2005), 81–108.

[15] Alborz Geramifard, Joshua Redding, Nicholas Roy, and Jonathan P. How. 2011. UAV cooperative control with stochastic risk models. In *Proceedings of the 2011 American Control Conference.* IEEE, 3393–3398.

[16] Antoine Girard. 2005. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control.* Springer, 291–305.

[17] Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. 2008. Safe exploration for reinforcement learning. In *ESANN.* 143–148.

[18] John Hertz, Anders Krogh, and Richard G. Palmer. 1991. *Introduction to the Theory of Neural Computation.* Addison-Wesley/Addison Wesley Longman.

[19] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2019. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Hybrid Systems: Computation and Control (HSCC).*

[20] Kyle D. Julian, Mykel J. Kochenderfer, and Michael P. Owen. 2018. Deep neural network compression for aircraft collision avoidance systems. *arXiv preprint arXiv:1810.04240* (2018).

[21] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. 2018. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC).* IEEE, 6059–6066.

[22] Kristofer D. Kusano and Hampton Gabler. 2011. Method for estimating time to collision at braking in real-world, lead vehicle stopped rear-end crashes for use in pre-crash system design. *SAE International Journal of Passenger Cars-Mechanical Systems* 4, 2011-01-0576 (2011), 435–443.

[23] David N. Lee. 1976. A theory of visual control of braking based on information about time-to-collision. *Perception* 5, 4 (1976), 437–459.

[24] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[25] Teodor Mihai Moldovan and Pieter Abbeel. 2012. Safe exploration in Markov decision processes. *arXiv preprint arXiv:1205.4810* (2012).

[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2574–2582.

[27] Sriram Sankaranarayanan, Souradeep Dutta, and Xin Chen. 2019. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Hybrid Systems: Computation and Control (HSCC).*

[28] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *Hybrid Systems: Computation and Control (HSCC).*

[29] Hoang-Dung Tran, Patrick Musau, Diego Manzanas Lopez, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. 2019. Parallelizable reachability analysis algorithms for feed-forward neural networks. In *7th International Conference on Formal Methods in Software Engineering (FormaliSE2019), Montreal, Canada.*

[30] Hoang-Dung Tran, Patrick Musau, Diego Manzanas Lopez, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. 2019. Star-based reachability analsysis for deep neural networks. In *23rd International Symposisum on Formal Methods (FM'19).* Springer International Publishing.

[31] Hoang-Dung Tran, Luan Viet Nguyen, Nathaniel Hamilton, Weiming Xiang, and Taylor T. Johnson. 2019. Reachability analysis for high-index linear differential algebraic equations (DAEs). In *17th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'19).* Springer International Publishing.

[32] Hoang-Dung Tran, Luan Viet Nguyen, Patrick Musau, Weiming Xiang, and Taylor T. Johnson. 2019. Decentralized real-time safety verification for distributed cyber-physical systems. In *Formal Techniques for Distributed Objects, Components, and Systems (FORTE'19),* Jorge A. Pérez and Nobuko Yoshida (Eds.). Springer International Publishing, Cham, 261–277.

[33] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. 2018. Simulation-based adversarial test generation for autonomous vehicles with machine learning components. *arXiv preprint arXiv:1804.06760* (2018).

[34] Weiming Xiang, Diego Manzanas Lopez, Patrick Musau, and Taylor T. Johnson. 2019. Reachable set estimation and verification for neural network models of nonlinear dynamic systems. In *Safe, Autonomous and Intelligent Vehicles.* Springer, 123–144.

[35] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. 2017. Reachable set computation and safety verification for neural networks with ReLU activations. *arXiv preprint arXiv:1712.08163* (2017).

[36] Weiming Xiang, Hoang-Dung Tran, Joel A. Rosenfeld, and Taylor T. Johnson. 2018. Reachable set estimation and safety verification for piecewise linear systems with neural network controllers. *arXiv preprint arXiv:1802.06981* (2018).

Received April 2019; revised June 2019; accepted July 2019