EPiC Series in Computing

Volume 61, 2019, Pages 201–210

ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems

Verification of Closed-loop Systems with Neural Network Controllers

Diego Manzanas Lopez, Patrick Musau, Hoang-Dung Tran, and Taylor T. Johnson

Vanderbilt University, Nashville, Tennessee, USA
diego.manzanas.lopez@vanderbilt.edu,patrick.musau@vanderbilt.edu,
dung.h.tran@vanderbilt.edu taylor.johnson@vanderbilt.edu

Abstract

This benchmark suite presents a detailed description of a series of closed-loop control systems with artificial neural network controllers. In many applications, feed-forward neural networks are heavily involved in the implementation of controllers by learning and representing control laws through several methods such as model predictive control (MPC) and reinforcement learning (RL). The type of networks that we consider in this manuscript are feed-forward neural networks consisting of multiple hidden layers with ReLU activation functions and a linear activation function in the output layer. While neural network controllers have been able to achieve desirable performance in many contexts, they also present a unique challenge in that it is difficult to provide any guarantees about the correctness of their behavior or reason about the stability a system that employs their use. Thus, from a controls perspective, it is necessary to verify them in conjunction with their corresponding plants in closed-loop. While there have been a handful of works proposed towards the verification of closed-loop systems with feed-forward neural network controllers, this area still lacks attention and a unified set of benchmark examples on which verification techniques can be evaluated and compared. Thus, to this end, we present a range of closed-loop control systems ranging from two to six state variables, and a range of controllers with sizes in the range of eleven neurons to a few hundred neurons in more complex systems.
Category: Academic **Difficulty: High**

Acknowledgement The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant number SHF 1736323, the Air Force Office of Scientific Research (AFOSR) through contract numbers FA9550-15-1-0258, FA9550-16-1-0246, and 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

1 Context and Origins.

In recent years, advances in Artficial Intelligence (AI) have enabled a diverse range of technologies that are directly impacting people's everyday lives [16]. Particularly, within this space, machine learning methods such as Deep Learning (DL) have achieved levels of accuracy and performance that are competitive or better than humans for tasks such as pattern and image recognition [12], natural language processing [7], and knowledge representation and reasoning [15,22]. Despite these achievements, there have been reservations about incorporating them into safety critical systems [11] due to their susceptibility to unexpected and errant behavior from a slight perturbation in their inputs [18]. Furthermore, neural networks are often viewed as "black boxes" since the underlying operation of the neuron activations is often indiscernible [22].

In light of these challenges, there has been significant work towards the creation of methods and verification tools that can formally reason about the behavior of neural networks [22]. However, the vast majority of these techniques have only been able to deal with feed-forward neural networks with piecewise-linear activation functions [4]. Additionally, the bulk of these methods have primarily considered the verification of input-output properties of neural networks in isolation [22], and there are only a handful of works that have explicitly addressed the verification of closed-loop control systems with neural network controllers [5, 8, 19–21]. One of the central challenges in verifying neural network control systems is that applying existing methodology to these systems is not straightforward [9], and a simple combination of verification tools for non-linear ordinary differential equations along with a neural network reachability tool suffers from severe overestimation errors [5]. Still, the verification of closed loop neural network systems is deeply important as they naturally arise in safety critical systems [5] such as autonomous vehicles, and complex control systems that make use of model predictive control and reinforcement learning [16]. Thus, there is a compelling need for methods and advanced software tools that can effectively deal with the complexities exhibited by these systems [5].

Inspired by a shortage of verification methods for closed-loop neural network control systems in the research literature, the central contribution of this paper is the provision of a set of executable benchmarks that have been synthesized using methods such as reinforcement learning [17], and model predictive control [14]. The problems elucidated in the paper are modeled using Simulink/Stateflow (SLSF) and are available at the following github repository[1]. We aim to provide a thorough problem description to which the numerous tools and approaches for non-linear systems and neural network verification present in the research community can be evaluated and compared [22]. If the research community is able to devise acceptable solutions to the aformentioned challenges they will stimulate the development of robust and intelligent systems with the potential to bring unparalleled benefits to numerous application domains.

2 Description of benchmarks.

In this manuscript, we present a set of linear and non-linear closed-loop systems with continuous-time plants and feedforward neural networks controllers trained using different controls schemes such as reinforcement learning or model predictive control (MPC). A typical architecture describing the structure of these systems is displayed in Figure 2.1. All the neural networks

[1] https://github.com/verivital/ARCH-2019

considered in this work possess a similar structure using the ReLU activation function in the hidden layers and use a linear activation function in the output layer. This structure makes them more amenable to the verification methods and tools available within the research literature [22].

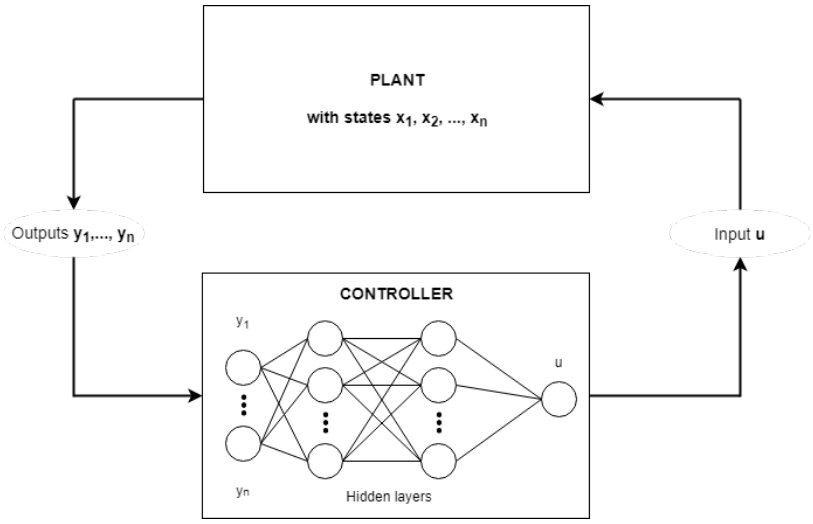

Figure 2.1: Illustration of a closed-loop control system with a neural network controller.

The benchmarks elucidated in this paper are modeled using Simulink/Stateflow and we have also used the Hybrid Source Transformer tool (HYST)[2] [1] to transform the models of the plants into the SpaceEx format [6]. Additionally, we have also provided the neural network controllers, in a variety of formats including a matlab format used in the (Neural Network Verication) NNV[3] framework proposed by Tran et al. [19], Simulink models, and the Open Neural Network Exchange[4] (ONNX) format. The following section presents a brief description of the benchmarks presented for verification.

2.1 Linear Inverted pendulum.

An inverted pendulum is a popular system commonly used as a benchmark in the area of control theory. This system consists of a pendulum attached to a cart that is pointing upwards, which is an unstable system in the absence of a controller. The verification goal of this benchmark is to show that eventually the pendulum stops at the upright position and stays there. The dynamics of this system have been linearized using the small angle approximation, for further details see [10].

$$\begin{aligned} \ddot{x} &= 0.0043\dot{\theta} - 2.75\theta + 1.94u - 10.95\dot{x}, \\ \ddot{\theta} &= 28.58\theta - 0.044\dot{\theta} - 4.44u + 24.92\dot{x}, \end{aligned} \tag{2.1}$$

[2] https://github.com/verivital/hyst
[3] https://github.com/verivital/nnv
[4] https://github.com/onnx

In eq. 2.1u is the input torque, x is the position and $\dot{x}$ is the velocity of the cart, θ is the angle and $\dot{\theta}$ is the angular velocity of the pendulum. Except for $\dot{\theta}$ which is unconstrained, we set the range of x to be between [-0.7, 0.7], the velocity $\dot{x}$ to be between [-1.0, 1.0], and θ to be between [-0.5236, 0.5236]. The initial conditions are set to 0 for the linear and angular velocities, $x_0 \in$ [-0.5, 0.5], and $\theta_0 \in$ [-0.2, 0.2]. The neural network controller has four inputs (x, $\dot{x}$, θ, $\dot{\theta}$), one output (u), and two layers with [10, 1] neurons. The verification goal is to show that the controller will drive the pendulum to its vertical position ($\theta = 0$), and cause it to remain there within 12 seconds.

2.2 Non-linear Cart-Pole.

This benchmark is obtained from the OpenAI gym [3], and corresponds to the version of the cart-pole introduced by Barto, Sutton, and Anderson [2]. Although the overall scheme of this system is very similar to the inverted pendulum, which also consists of a pendulum attached to a cart, this variation of the system is unique in that the dynamic equations used to represent it are more complex. Moreover, in this case, we represent the dynamics of the system as a set of non-linear differential equations and alter the controller structure in addition to the initial states and constraints. The dynamics of the cart-pole system are described as follows

$$\begin{aligned} \ddot{x} &= \frac{u + ml\omega^2 \sin(\theta)}{m_t} - \frac{ml(g\sin(\theta) - \cos(\theta))(\frac{u+ml\omega^2\sin(\theta)}{m_t})}{l(\frac{4}{3} - m\frac{\cos^2(\theta)}{m_t})}\frac{\cos(\theta)}{m_t}, \\ \ddot{\theta} &= \frac{g\sin(\theta) - \cos(\theta)(\frac{u+ml\omega^2\sin(\theta)}{m_t})}{l(\frac{4}{3} - m\frac{\cos^2(\theta)}{m_t})}\frac{\cos(\theta)}{m_t} \end{aligned} \tag{2.2}$$

where $u \in \{-10, 10\}$ is the input force, which either pushes the cart left or right, $g = 9.8$ is gravity, $m = 0.1$ is the pole's mass, $l = 0.5$ is half the pole's length, $m_t = 1.1$ is the total mass, x is the position of the cart, θ is the angle of the pendulum with respect to the positive y-axis, $v = \dot{x}$ is the linear velocity of the cart, and $\omega = \dot{\theta}$ is the angular velocity of the pendulum. The controller has four inputs $(x, \dot{x}, \theta, \dot{\theta})$, four layers with $[24, 48, 12, 2]$ neurons respectively, and two outputs. The two outputs are then compared, and the input sent to the plant depends on which output index has the greatest value. Thus, as an example if $output_1 > output_2$ then the input force supplied to the plant is 10. However if $output_1 < output_2$ then the input supplied to the plant is -10.

For this benchmark, the verification objective is to demonstrate that the pole will never be more than 15 degrees from the vertical positions, and it will eventually reach the upward position and that it will remain there. Similar to the inverted pendulum, it needs to be verified that these properties are satisfied in the first 12 seconds. In other words, the goal is to achieve a value of $\theta = 0$ and stay there. The initial conditions for all state variables were chosen uniformly at random between [-0.05, 0.05], but we did not impose any other constraints on the system.

2.3 Acrobot.

Similarly, this benchmark was obtained from OpenAI gym [3], and it was originally proposed as a benchmark for reinforcement learning. The Acrobot system consists of two linked pendulums with only the second joint actuated. Both links can swing freely and can pass by each other, i.e., they don't collide when they have the same angle. The goal is to swing the bottom link at a height at least the length of one link above the base, which can be formally defined as

changing θ_1 and θ_2 so that $-\cos(\theta_1)$ - $\cos(\theta_1+\theta_2) \geq 1$. The dynamics of the Acrobot can be expressed by the following set of equations

$$\begin{aligned}
d_1 &= m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(\theta_2)) + I_1 + I_2,\\
d_2 &= m_2(l_{c2}^2 + l_1 l_{c2}\cos(\theta_2)) + I_2,\\
\phi_1 &= -m_2 l_1 l_{c2}\dot{\theta}_2^2 \sin(\theta_2) - 2m_2 l_1 l_{c2}\dot{\theta}_2\dot{\theta}_1 \sin(\theta_2) + (m_1 l_{c1} + m_2 l_1)g\cos(\theta_1 - \frac{\pi}{2}) + \phi_2,\\
\phi_2 &= m_2 l_{c2} g\cos(\theta_1 + \theta_2 - \frac{\pi}{2}),\\
\ddot{\theta}_1 &= -\frac{d_2\ddot{\theta}_2 + \phi_1}{d_1},\\
\ddot{\theta}_2 &= \frac{\tau + \frac{d_2}{d_1}\phi_1 - \phi_2}{m_2 l_{c2}^2 + I_2 - \frac{d_2^2}{d_1}},
\end{aligned} \tag{2.3}$$

where $m_1 = m_2 = 1$ are the masses of both links, $l_1 = l_2 = 1$ are the lengths of the links, l_{c1} $= l_{c2} = 0.5$ are the positions of the center or mass of both links, τ is the input torque, and g $= 9.8$ is gravity. The outputs of this systems are $\cos\theta_1$, $\sin\theta_1$, $\cos\theta_2$, $\sin\theta_2$, $\dot{\theta}_1$ and $\dot{\theta}_2$, which are the inputs of the neural network containing [15,15,3] neurons in its 3 layers. This neural network works as a classification function (similar to Cart-Pole), which outputs either 1, 0 or -1 depending on the index of the greatest output value of the neural network.

We impose constraints on the maximum angular velocities allowed in the system $v_{max1} = 4\pi$ and $v_{max2} = 9\pi$, for links 1 and 2 respectively. The initial states of the system θ_1 and θ_2 are initialized uniformly at random to be within [-0.1, 0.1], and their corresponding rate of change, ω_1 and ω_2, are initialized uniformly at random between [-0.1, 0.1] as well. The verification objective of this system is to prove that the bottom link reaches, at east once, a height at least the length of one link above the base with in the first 20 seconds. Formally, the systems needs to satisfy the following formula:

$$-\cos(\theta_1) - \cos(\theta_1 + \theta_2) \geq 1 \tag{2.4}$$

2.4 Mountain Car.

This benchmark was obtained from the OpenAI gym [3], as reinforcement learning benchmark, although it was first introduced by Andrew Moore in his PhD thesis [13]. However, we present a continuous version of it, similar to the one evaluated by Ivanov et al. [9]. The overall system consists of a car located on a one-dimensional track between two mountains and the objective of the car is to climb up the mountain on the right. However, the car's engine has limited power and it cannot scale the mountain in a single pass. Therefore, the only way for the car to achieve its goal is to drive back to the mountain to the left to build up momentum and then climb up the mountain to the right. The controller has been trained using reinforcement learning, and the dynamics of the system are given as follows* [5]

$$\begin{aligned}
\dot{p} &= v,\\
\dot{v} &= 0.0015u + 0.0025\cos(3p)
\end{aligned} \tag{2.5}$$

[5]Originally introduced in discrete-time (time step not specified) in https://link.springer.com/content/pdf/10.1007$%$2FBF00114726.pdf

where p is the position and v is the velocity of the car. The controller presented in this system is a neural network with 2 inputs (p and v), 3 layers with [16, 16, 1] neurons respectively, and 1 output (u). The verification objective of this benchmark is to show that the car will reach the top of the right mountain before the first 50 seconds of execution, which can be formally described as driving the car to $p \geq 0.45$. This system has constraints on the velocity, which needs to be within [-0.07, 0.07] and on the position, which cannot be outside [-1.2, 0.6]. The initial conditions are defined to be $\mathrm{v}_0 = 0$, and $p_0 \in$ [-0.6, -0.4].

2.5 MPC Quadrotor.

In this case study, we present a six-dimensional control-affine model of a quadrotor controlled by a feed-forward neural network initially introduced in [9], although here we have modified the controller to have only ReLU and linear activation functions. The goal of this benchmark is to verify that the quadrotor will reach its final position without colliding with nearby objects. Formally, the verification problem is to ensure that the quadrotor follows the path planner and tries to stay as close as possible to the path, i.e., stabilize the system of relative states $r := q - p$. Specifically, we want the distance from the quadrotor to the path to be under 0.32 meters for at least 10 seconds. The dynamics of this system are described as follows:

$$\dot{q} := \begin{pmatrix} \dot{p_x^q} \\ \dot{p_y^q} \\ \dot{p_z^q} \\ \dot{v_x^q} \\ \dot{v_y^q} \\ \dot{v_z^q} \end{pmatrix} = \begin{pmatrix} v_x^q \\ v_y^q \\ v_z^q \\ g\tan\theta \\ -g\tan\phi \\ \tau - g \end{pmatrix}, \dot{p} := \begin{pmatrix} \dot{p_x^p} \\ \dot{p_y^p} \\ \dot{p_z^p} \\ \dot{v_x^p} \\ \dot{v_y^p} \\ \dot{v_z^p} \end{pmatrix} = \begin{pmatrix} b_x \\ b_y \\ b_z \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.6}$$

where $\dot{\mathrm{q}}$ and $\dot{\mathrm{p}}$ are the quadrotor and planner dynamics, respectively; v_i and p_i represents the velocity and position of coordinate i respectively, where $i \in \{x, y, z\}$. θ, ϕ, and τ are the control inputs (pitch, roll and thrust), $\mathrm{g} = 9.81\ \mathrm{m}/s^2$ is gravity, and b_i are piecewise constants. In this case, we have constraints on the inputs which can only take on two possible values each: θ, ϕ $\in \{-0.1, 0.1\}$, and $\tau \in \{7.81, 11.81\}$.

Some further constraints are presented in the initial states, where $(p_x^r,\ p_y^r) \in [-0.05, -0.05] \times [0.05, 0.05]$. As a controller, we use a neural network with 6 inputs (p_x^r, p_y^r, p_z^r, v_x^r, v_y^r, v_z^r), 3 layers with [20, 20, 8] neurons respectively, and 8 outputs. This is similar to the neural network architectures seen in the acrobot and cart-pole, as the neural network has a predetermined set of output actions, eight in this case, dependent on the index of the greatest output value. In each case a unique combination of θ, ϕ and τ is sent to the controller.

2.6 Adaptive Cruise Controller (ACC).

The Adaptive Cruise Control (ACC) System simulates a system that tracks a set velocity and maintains a safe distance from a lead vehicle by adjusting the longitudinal acceleration of an ego vehicle. The neural network computes optimal control actions while satisfying safe distance, velocity, and acceleration constraints using model predictive control (MPC) [14]. For this case study, the ego car is set to travel at a set speed $V_{set} = 30$ and maintains a safe distance D_{safe} from the lead car. The car's dynamics are described as follows:

$$\begin{aligned} \dot{x}_{\mathrm{lead}}(t) = v_{\mathrm{lead}}(t), \dot{v}_{\mathrm{lead}}(t) = \gamma_{\mathrm{lead}}(t), \dot{\gamma}_{\mathrm{lead}}(t) = -2\gamma_{\mathrm{lead}}(t) + 2a_{\mathrm{lead}} - uv_{\mathrm{lead}}^2(t), \\ \dot{x}_{\mathrm{ego}}(t) = v_{\mathrm{ego}}(t), \dot{v}_{\mathrm{ego}}(t) = \gamma_{\mathrm{ego}}(t), \dot{\gamma}_{\mathrm{ego}}(t) = -2\gamma_{\mathrm{ego}}(t) + 2a_{\mathrm{ego}} - uv_{\mathrm{ego}}^2(t), \end{aligned} \tag{2.7}$$

where x_i is the position, v_i is the velocity, γ_i is the acceleration of the car, a_i is the acceleration control input applied to the car, and $u = 0.0001$ is the friction control where $i \in \{\text{ego, lead}\}$. For this benchmark we have developed four neural network controllers with 3, 5, 7, and 10 hidden layers of 20 neurons each. All of them have the same number of inputs (v_{set}, T_{gap}, v_{ego}, D_{rel}, v_{rel}), and one output (a_{ego}). The verification objective of this system is that given a scenario where both cars are driving safely, the lead car suddenly slows down with $a_{\text{lead}} =$ -2. We want to check whether there is a collision in the following 5 seconds. Formally, this safety specification of the system can be expressed as $D_{\text{rel}} = x_{\text{lead}}$ - $x_{\text{ego}} \geq D_{\text{safe}}$, where $D_{\text{safe}} = D_{\text{default}} + T_{\text{gap}} \times v_{\text{ego}}$, and $T_{\text{gap}} = 1.4$ seconds and $D_{\text{default}} = 10$. The initial conditions are: $x_{\text{lead}}(0) \in [90,110]$, $v_{\text{lead}}(0) \in [32,32.2]$, $\gamma_{\text{lead}}(0) = \gamma_{\text{ego}}(0) = 0$, $v_{\text{ego}}(0) \in [30, 30.2]$, $x_{\text{ego}} \in [10,11]$. See Figure 3.1 for a verification example using this ACC benchmark.

2.7 Other benchmarks.

Finally, we also present a set of benchmark problems originally presented by Dutta et al. in [5]. In their work, they present a collection of ten continuous dynamical systems, including a quadrotor and a car. However, our benchmark suite only considers benchmarks 1 through 8. Similar to the previous examples, all of these benchmarks contain feed-forward neural network controllers with ReLU activation functions in the hidden layers and a linear function in the output layer. In thier work the authors trained all of the neural networks using a MPC scheme and included a disturbance w (except benchmark 8) to the control input u. We refer readers to the following paper for a more detailed description of this set of benchmarks. [5].

3 Reachability Analysis.

In this section we present a brief investigation of the verification of the Adaptive Cruise Controller (ACC) Benchmark in order to demonstrate the verification of a closed loop neural network control system. This experiment was originally proposed in [19] and we present the results here as an example of the methods currently available within the research literature. As previously mentioned, the safety verification problem of interest is that when the ego car makes use of the speed control mode and both the lead car and the ego car are driving with a safe distance between them, if the lead car suddenly brakes, we expect that the neural network controllers will ensure that the ego car brakes correspondingly so as to maintain a safe distance between the two cars [19]. In this particular case we consider a neural network controller with 5 hidden layers each consisting of 20 neurons and make use of the Neural Network Verification Toolbox (NNV) proposed by Tran et al. [19] to perform the verification. The initial conditions and safety specifications are as described in subsection 2.6. The verification results demonstrate that aforementioned controller is safe since it guarantees that the ego car maintains a safe distance for the range of the lead car's initial position that we considered as shown in Figure 3.1. Utilizing a computer with the following configuration: Intel Core i7-6700 @ 3.4 GHZ ×8 Processor, 62 GiB Memory, 64-bit Ubuntu 16.04.3 LTS OS. we were able to verify this safety property with an average execution time of 305.17 seconds. The experiments can be found in the following repository: https://github.com/verivital/nnv.

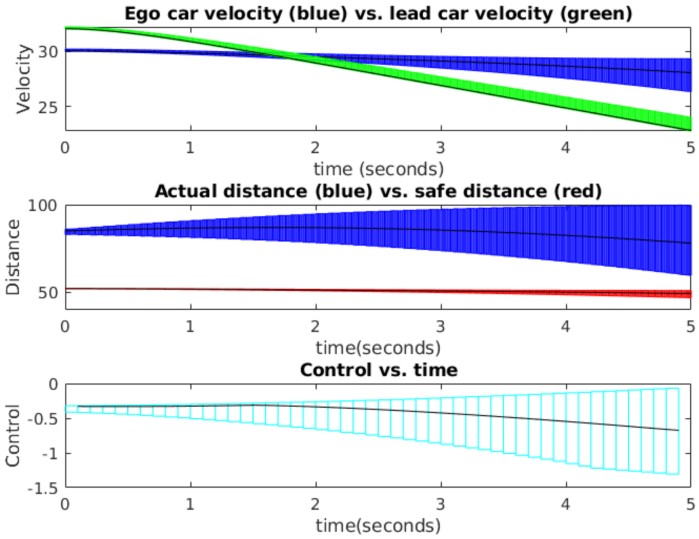

Figure 3.1: Verification results for the ACC Benchmark. In all plots the black line represents a simulation trace of the system. Top subplot: The green area represents the lead car's velocity reachable set, and the blue area represents the ego car's velocity reachable set. Middle subplot: The blue area represents the reachable set of the distance between the ego car and the lead car. The red area represents the threshold under which the system becomes unsafe. Bottom subplot: The light blue area represents the reachable set of the neural network control input to the plant model.

4 Outlook and Conclusion.

In this manuscript we have presented a diverse set of challenging benchmarks for the verification of closed-loop control systems with neural network controllers. The interest in these problems is that they seek to stimulate the creation of methods and software tools that can handle the unique challenges, such as severe overestimation errors, encountered by considering the verification of such systems. Currently only a handful of methods have been proposed at solving this problem and the existing verification methods for neural networks in isolation cannot be leveraged in a straightforward fashion with tools designed for the verification of non-linear plant models. Neural networks arise naturally in safety critical systems as they are a powerful paradigm for solving deeply complex control problems and if the research community can demonstrate an ability to effectively and efficiently reason about the correctness of their behavior within a controls system context, they will stimulate the development of robust intelligent systems with the potential to bring unparalleled benefits to a diversity of application domains. The neural networks presented in this manuscript are all feed-forward networks using the ReLU activation function in the hidden layers and a linear activation function in the output layer. In future work we wish to consider control systems utilizing Convolutional Neural Networks as well as network architectures that make use of more general activation functions.

References

[1] Bak, S., Bogomolov, S., and Johnson, T. T. Hyst: A source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control* (New York, NY, USA, 2015), HSCC '15, ACM, pp. 128–133.

[2] Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics SMC-13*, 5 (Sep. 1983), 834–846.

[3] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR abs/1606.01540* (2016).

[4] Bunel, R., Turkaslan, I., Torr, P. H. S., Kohli, P., and Kumar, M. P. Piecewise linear neural network verification: A comparative study. *CoRR abs/1711.00455* (2017).

[5] Dutta, S., Chen, X., and Sankaranarayanan, S. Reachability analysis for neural feedback systems using regressive polynomial rule inference. *Hybrid Systems: Computation and Control (HSCC)* (2019).

[6] Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O. Spaceex: Scalable verification of hybrid systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV)* (2011), S. Q. Ganesh Gopalakrishnan, Ed., LNCS, Springer.

[7] Goldberg, Y. A primer on neural network models for natural language processing. *CoRR abs/1510.00726* (2015).

[8] Ivanov, R., Weimer, J., Alur, R., Pappas, G. J., and Lee, I. Verisig: verifying safety properties of hybrid systems with neural network controllers. *CoRR abs/1811.01828* (2018).

[9] Ivanov, R., Weimer, J., Alur, R., Pappas, G. J., and Lee, I. Verisig: verifying safety properties of hybrid systems with neural network controllers. *CoRR abs/1811.01828* (2018).

[10] Johnson, T. Stability analysis of simplex architecture controlled inverted pendulum.

[11] Katz, G., Barrett, C. W., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR abs/1702.01135* (2017).

[12] Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. A survey of the recent architectures of deep convolutional neural networks. *CoRR abs/1901.06032* (2019).

[13] Moore, A. W. Efficient memory-based learning for robot control.

[14] Qin, S. J., and Badgwell, T. A. An overview of nonlinear model predictive control applications. In *Nonlinear Model Predictive Control* (Basel, 2000), F. Allgöwer and A. Zheng, Eds., Birkhäuser Basel, pp. 369–392.

[15] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature 529* (2016), 484–503.

[16] Stone, P., Brooks, R., Brynjolfsson, E., Calo, R., Etzioni, O., Hager, G., Hirschberg, J., Kalyanakrishnan, S., Kamar, E., Leyton-Brown, K., Parkes, D. C., Press, W., Saxenian, A., Shah, J., Tambe, M., and Teller, A. "artificial intelligence and life in 2030." one hundred year study on artificial intelligence: Report of the 2015-2016 study panel, 2016.

[17] Sutton, R. S., and Barto, A. G. *Introduction to Reinforcement Learning*, 1st ed. MIT Press, Cambridge, MA, USA, 1998.

[18] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR abs/1312.6199* (2013).

[19] Tran, H.-D., Yang, X., Lopez, D. M., Musau, P., Nguyen, L. V., Xiang, W., and Johnson, T. T. Nnv: A neural network verification tool for autonomous cyber-physical systems. *Computer-Aided Verification (CAV)* (2019).

[20] Tuncali, C. E., Kapinski, J., Ito, H., and Deshmukh, J. V. Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. *CoRR abs/1804.03973* (2018).

[21] Xiang, W., and Johnson, T. T. Reachability analysis and safety verification for neural network control systems. *CoRR abs/1805.09944* (2018).

[22] Xiang, W., Musau, P., Wild, A. A., Lopez, D. M., Hamilton, N., Yang, X., Rosenfeld, J. A., and Johnson, T. T. Verification for machine learning, autonomy, and neural networks survey. *CoRR abs/1810.01989* (2018).