Verification of Continuous Time Recurrent Neural Networks (Benchmark Proposal)

Patrick Musau and Taylor T. Johnson

Vanderbilt University, Nashville, Tennessee, USA
patrick.musau@vanderbilt.edu, taylor.johnson@vanderbilt.edu

Abstract

This manuscript presents a description and implementation of two benchmark problems for continuous-time recurrent neural network (RNN) verification. The first problem deals with the approximation of a vector field for a fixed point attractor located at the origin, whereas the second problem deals with the system identification of a forced damped pendulum. While the verification of neural networks is complicated and often impenetrable to the majority of verification techniques, continuous-time RNNs represent a class of networks that may be accessible to reachability methods for nonlinear ordinary differential equations (ODEs) derived originally in biology and neuroscience. Thus, an understanding of the behavior of an RNN may be gained by simulating the nonlinear equations from a diverse set of initial conditions and inputs, or considering reachability analysis from a set of initial conditions. The verification of continuous-time RNNs is a research area that has received little attention and if the research community can achieve meaningful results in this domain, then this class of neural networks may prove to be a superior approach in solving complex problems than other network architectures.
Category: Academic **Difficulty:** High

1 Context and Origins

Artificial Neural Networks have demonstrated an effective and powerful ability to achieve success in numerous contexts, such as adaptive control [35], autonomous vehicles, evolutionary robotics, pattern recognition, image classification, and nonlinear system identification and control [30] [16]. Despite this success, there have been reservations in incorporating them into safety critical systems [21] due to their susceptibility to unexpected and errant behavior by a slight perturbation in their inputs and initial conditions [34] [29]. Typically, neural networks are viewed as "black boxes" since the underlying operation of the neuron activations is often indiscernible to the creators of the network [8]. In light of these challenges there has been significant work towards obtaining formal guarantees about the behavior of neural networks [3]. However, the majority of verification schemes have only been able to deal with neural networks that make use of piecewise-linear activation functions [6]. This is due to the great difficulty exhibited in obtaining formal guarantees about even simple properties of neural networks. In fact, neural network verification has been demonstrated to be an NP-complete problem and

while techniques that make use of satisfiability modulo theories [28], mixed integer programming [33], robustness testing [4], and linear programming [11] [29] have been able to deal with small networks, they are incapable of dealing with the complexity and scale of the majority of networks present in real-life applications [21]. Moreover, the majority of verification approaches have dealt only with feed-forward and convolutional neural network architectures.

One class of neural networks that has received particularly little attention in the verification literature is the class of recurrent neural networks. Whereas both feed-forward networks and recurrent networks have demonstrated an ability to approximate continuous functions to any accuracy [14], recurrent neural networks have exhibited several advantages over their feed-forward counterparts [23]. By allowing for the presence of feedback connections in their architecture, recurrent neural networks are able to retain information about the past and capture a higher degree of sophisticated dynamics using fewer neurons than their feed-forward counterparts [5]. In fact, recurrent neural networks have demonstrated a higher level of success in solving problems in which there is a temporal relation between events [26] such as capturing the behavior of biological neurons [24], dynamical system identification [20], and speech recognition [1]. Therefore, they represent a more attractive framework than feed-forward networks in these domains [39]. However, due to the complexity exhibited by their architecture as well as the non-linear nature of their activation functions the verification approaches currently available in the research literature are incapable of being applied to these networks. Thus, there is an immediate need for methods and advanced software tools that can provide formal guarantees about their operation [21], particularly in the context of the system identification and the control of safety critical systems.

In light of this shortcoming, the following paper presents two benchmark problems for the verification of a specific class of recurrent neural networks known as continuous-time recurrent neural networks (CTRNNs). Since the dynamics of CTRNNs can be expressed solely by a set non-linear ordinary differential equations (ODEs), the verification of such systems relies on an ability to reason about the reachable set from a set of initial conditions and inputs [31]. The two CTRNN benchmark problems we present are described as follows: the first is a network without inputs for the approximation of a fixed point attractor described in [38], and the second deals with a CTRNN used for the identification of a damped forced pendulum as described in [10]. The problems elucidated in the paper are modeled using Simulink/Stateflow (SLSF), Matlab scripts, and are available in the SpaceEx format[1] [13]. We aim to provide a thorough problem description to which the numerous tools and approaches for non-linear systems present in the research community can be evaluated and compared [31]. This paper serves as a first step towards recurrent neural network verification.

2 General Mathematical Model for Continuous Time Recurrent Neural Networks

The dynamics of a continuous-time recurrent neural network with n neurons is given by the following system of ordinary differential equations:

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_{j=1}^{n} w_{ij} f(u_j(t) + \theta_i) + I_i(t) \tag{2.1}$$

[1]The benchmark models are available at https://bitbucket.org/verivital/ctrnnbenchmark

where: $u_i(t)$ denotes the internal state of the i-th neuron ($i = 1 \ldots, n$), θ_i denotes a bias term for the i-th neuron, τ_i denotes the time constant, w_{ij} denotes the connection strength between the i-th and j-th neurons, $I_i(t)$ denotes the input to the i-th neuron, and $f(u_j(t))$ is the output of the j-th neuron [17]. The function f is typically referred to as the activation function of a neuron, and in most cases it is typically either the logistic sigmoid function or the hyperbolic tangent function given by equations (2.2) and (2.3) respectively.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$f(x) = tanh(x) \tag{2.3}$$

Equation (2.1) can be recast into matrix form as:

$$\dot{\vec{u}}(t) = -\frac{1}{\tau}\vec{u}(t) + \mathbf{W}f(\vec{u}(t) + \theta_i) + \vec{I}(t) \tag{2.4}$$

where: $\vec{u}(t) = (u_1 \ldots u_n)^T$ and $\vec{I}(t)$ are n dimensional column vectors representing the neuron states and inputs respectively, $\mathbf{W} = (w_{ij})$ is a $n \times n$ weight matrix, and $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is an activation function mapping given by $f(\vec{u}(t)) = (f(u_1) \ldots f(u_n))$ [17].

In order to train a CTRNN to approximate the finite time trajectories of a dynamical system, the number of neurons n must be greater than the dimensionality of the function that we wish to approximate [18]. Typically the weights, biases, and time constants in equation (2.4) are modified using Genetic Algorithms [27] such as back-propagation through time, real-time recurrent learning, the extended Kalman filter, phase space learning, and reservoir computing [38]. In these schemes, the network is trained by minimizing the prediction error between the network output and dynamical system output on a set of time series data collected from the dynamical system. A common learning architecture is displayed in Figure 2.1. The algorithms used in training CTRNNS are quite complex and difficult to implement, however they have largely been successful in training networks to perform various tasks. An in-depth discussion of these techniques is beyond the scope of this paper and we refer readers to the following paper for further detail [27].

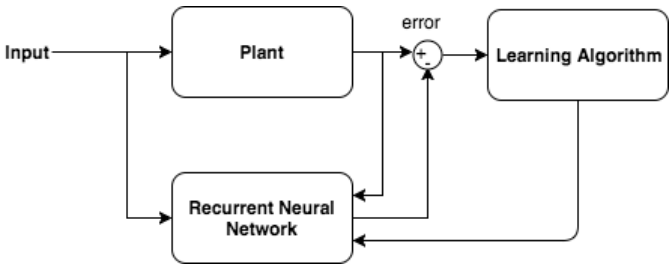

Figure 2.1: Diagram describing the architecture used to train a neural network for system identification

3 Description of Benchmarks

In this section, we present a description of the two benchmark problems proposed for verification. The first of these problems is a CTRNN with no inputs that captures the vector field dynamics of a fixed point attractor. The approximation of attractors represents a class of problems in recurrent neural network research known as fixed point learning [38] and networks

trained in this domain are typically associated with constraint satisfaction and associative memory [27]. The second problem describes the use of a CTRNN in the system identification of a forced damped pendulum. An accurate identification of the plant model is a crucial part in designing robust controllers [19] and in this problem [10], once the network has successfully been trained it is used calculate a linearized state feedback for PID control.

3.1 Approximation of Fixed Point Attractor

In dynamical systems, a fixed point attractor is a point in the phase space where the trajectories of a system converge to as time evolves from a given set of initial states [36]. For our problem we consider a two-dimensional system with a fixed point located at the origin [38]. The vector field for the attractor is given by the following function

$$F(x, y) = (a(x - p_1), b(x - p_2)) \tag{3.1}$$

where $p_1 = p_2 = 0$, $a = -0.2$, and $b = -0.3$. To approximate the behavior of this vector field during a 40 second time period, Adam Trischler et al. used two layer CTRNN that consisted of a single hidden layer with $m = 3$ hidden neurons and an output layer with $n = 2$ output neurons [38]. The recurrent connections are contained within the hidden layer as shown by Figure 3.1. The CTRNN was trained using an algorithm proposed in [38] in which a feed-forward

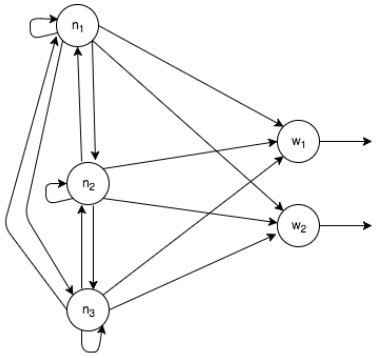

Figure 3.1: CTRNN architecture used in approximation of fixed point attractor

representation of the recurrent neural network is trained using back-propagation and then transformed back into a recurrent neural network. Thus, the differential equations specifying the dynamics of the CTRNN are expressed a little differently than in equation 2.1. They are given by:

$$\dot{\vec{u}} = -\frac{1}{\tau}\vec{u} + \mathbf{W}f(\vec{u}) \tag{3.2}$$

$$\vec{u}(0) = (\vec{q}(0), \mathbf{B}\vec{q}(0) + \vec{\theta})^T \tag{3.3}$$

where: $\vec{q}(0)$ is the initial condition for the fixed point attractor, $\mathbf{B}$ is a $m \times n$ matrix representing the connections among the hidden neurons, $\tau = 10^6$ is the time constant for each neuron, $\vec{\theta}$ represents the bias term for the neurons in the hidden layer, and $f(\vec{u}) = tanh(\vec{u})$ is the activation function used for each neuron [38].[2] The weight matrix $\mathbf{W}_{(n+m)\times(n+m)}$ used in

[2]The time constant τ determines the activation speed of a particular neuron. A description of the influence of time constants can be found in [40]

equation (6) is constructed using two matrices $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A}$ is an $n \times m$ matrix representing the connections from the hidden neurons to the output neurons and $\mathbf{B}$ is the matrix described as above. The matrices $\mathbf{W}$, $\mathbf{A}$, $\mathbf{B}$, and bias vector $\vec{\theta}$ trained on the vector field in equation 3.1 are given by,

$$\mathbf{W} = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{0} & \mathbf{BA} \end{bmatrix} \mathbf{A} = \begin{bmatrix} -1.20327 & -0.07202 & -0.93635 \\ 1.18810 & -1.50015 & 0.93519 \end{bmatrix} \mathbf{B} = \begin{bmatrix} 1.21464 & -0.07202 \\ 1.18810 & -1.50015 \\ 1.18810 & -1.50015 \end{bmatrix}$$

$\vec{\theta} = (-7.56499 \times 10^{-5}, 1.34708 \times 10^{-4}, -6.24925 \times 10^{-6})^T$. Since the network does not receive any inputs its trajectory is entirely dependent on the choice of initial conditions $\vec{q}(0)$. Simulating the network is computationally inexpensive and largely depends on the choice of numerical method used to discretize equation (3.2).

3.2 System Identification for Forced Damped Pendulum

The second problem we present is the approximation of a forced damped pendulum described by the second-order differential equation:

$$ml^2\ddot{\phi}(t) + v\dot{\phi}(t) + mgl\ sin\ \phi(t) = u(t) \tag{3.4}$$

where $m = 2.0\ kg$ is the mass of the object suspended by the pendulum, $v = 1.0$ kg m^2/s is the damping coefficient, $l = 1$ m is the length of pendulum and $u(t)$ is the driving signal at time t [10]. To approximate the trajectories of this system, A. Delgado et al. designed a CTRNN with a single hidden layer that consisted of five neurons whose dynamics are described by a set of equations that are an adaptation of equation (2.1). Since the pendulum is driven by a driving signal $u(t)$ the CTRNN ODE model must include the influence inputs. However, the activation of the neurons in the network may be influenced differently by the driving input signal [9] [25], and to capture this reality, the authors in [10] incorporated a vector representing the connection weights from the scalar input to each neuron denoted by $\vec{\Gamma}$. Additionally, the authors did not incorporate any biases in their CTRNN model. The dynamics of their model is described by:

$$\dot{\vec{u}}(t) = -\frac{1}{\tau}\vec{u}(t) + \mathbf{W}f(\vec{u}(t) + \theta_i) + \vec{\Gamma}u(t) \tag{3.5}$$

where $\vec{\Gamma}$ is an $n \times 1$ column vector denoting the input connection weights and $u(t)$ is the scalar input signal. The authors trained the network using time series data that was collected by supplying the pendulum model with random noise input data over a time period of 20 seconds. The performance of the network was optimized using a chemotaxis learning algorithm [10] and after training the network parameters are given by:

$$\mathbf{W} = \begin{bmatrix} 0.4684 & -2.4994 & 0.4211 & -0.2848 & 0.1995 \\ 1.3615 & 0.0642 & 0.0413 & -1.8925 & -1.6608 \\ -0.8185 & -0.9241 & -0.0743 & -0.1264 & 0.1484 \\ -0.3257 & 1.2319 & -1.0997 & 0.2192 & -0.8547 \\ -1.2444 & 0.4396 & -0.5466 & 1.7342 & -0.5953 \end{bmatrix}$$

$\vec{\Gamma} = (-0.005, -0.2111, 0.1689, 0.0645, -0.0413)^T$ $\vec{u}(0) = (-0.0097, -0.0065, 0.0171, -0.0097, 0.0025)^T$

4 Simulation of Models

We begin this section by discussing the results of simulating the CTRNN models presented in section 3 in order to assess their fidelity in capturing the dynamical systems that they represent. Figure 4.1 displays a 20 second simulation of the fixed point attractor system and its CTRNN representation from a set of 100 initial points bounded by $(x, y) \in [-1, 1]^2$. The results of the simulation demonstrated that the CTRNN was able to faithfully capture the dynamics of equation (3.1). In fact, the maximum separation between the trajectories of the neural network $\vec{y}_{nn}(t) \in \mathbb{R}^2$, and the fixed point attractor $\vec{y}_{fpa}(t) \in \mathbb{R}^2$, on any of the 100 paths that we considered was $||\vec{y}_{nn}(t) - \vec{y}_{fpa}(t)||_2 = 9.57 \times 10^{-3}$, as shown by Figure 4.3. Moreover, as the trajectories of the CTRNN model approached the fixed point, the output of all the neurons converged to zero causing the evolution of the network to halt. This indicates stability within the region that we considered.

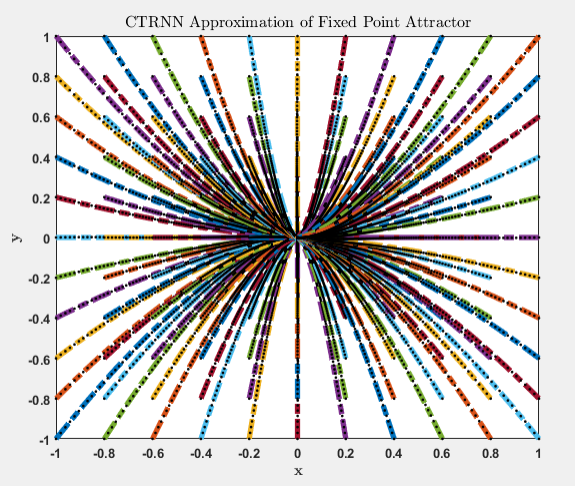

Figure 4.1: Trajectories of the fixed point attractor and CTRNN for various initial points $(x, y) \in [-1, 1] \times [-1, 1]$. The black dots correspond to the outputs of the neural network at various time instants while the colored lines represent the orbits of equation (3.1)

In a similar fashion, the CTRNN model of the forced damped pendulum displayed a high level of model fidelity in representing a chaotic system [15]. The results of several simulations using a series of sinusoidal input functions given by:

$$u(t) = c_1(\frac{\pi}{2})sin(\frac{2\pi t}{2.5}) + c_2(\frac{\pi}{2})sin(\frac{2\pi t}{5.0}) \tag{4.1}$$

is displayed in Figure 4.2. The maximum separation at any moment between the trajectories of the network model and the pendulum system was $|y_{nn}(t) - y_{dfp}(t)| = 2.16 \times 10^{-3}$, as shown in Figure 4.3 . Thus although both models are low dimensional neural networks, they are a good representation of the dynamical systems that they identify.

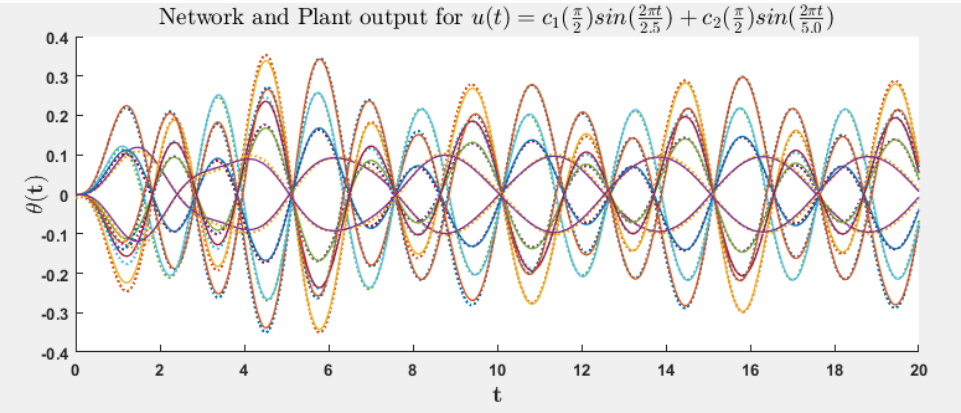

Figure 4.2: Four second simulation of the forced damped pendulum system and CTRNN

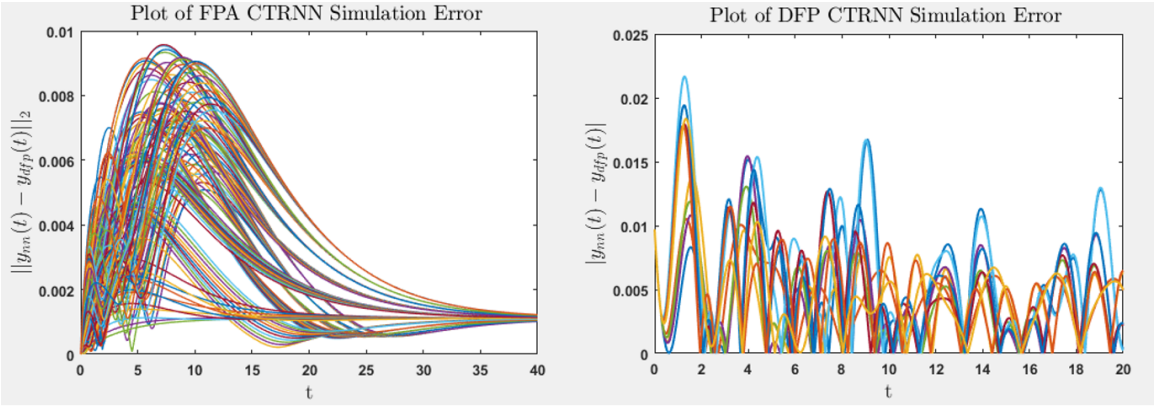

Figure 4.3: Simulation Error Plots for both CTRNN Models

5 Observations

This is a novel research direction. To the best of the authors' knowledge there are currently no verification approaches that deal with recurrent neural networks despite their success in numerous contexts. The development of successful verification approaches will have a monumental impact in the field of artificial intelligence, and in particular, in the design of neural network control systems. Thus, the deliberate consideration of this problem by the research community is considerably important.
Scalability: The number of variables in equation (2.1) increases linearly with the number of neurons present in the network. However the number of connections among the neurons increases quadratically and may lead to very large weight matrices. A typical real-world neural network can contain hundreds of thousands to millions of neurons [22], thus an effective verification approach will need to scale well with the dimensionality of large networks.
Challenges: Verifying continuous non-linear systems is a particularly difficult problem. In

particular, the performance of verification tools is dependent on the stiffness of the ODEs [32]. In our experiments we attempted to generate reachtubes for the CTRNN models using Flow* [7] and C2e2 [12], however both tools were not able to deal with the complexity exhibited by our models. This further serves to demonstrate the need for novel reachability tools for CTRNNS.

Figure 5.1: Stateflow Model for Damped Forced Pendulum CTRNNs

6 Outlook and Model Extensions

In this manuscript, we have presented a description and implementation of two challenging continuous-time recurrent neural network benchmark problems. The interest in these problems is that they serve as a first step in considering the verification of recurrent neural networks. The verification of CTRNNs has not been studied extensively and in this domain, the verification problem is reduced to an ability to reason about the reachable set of the non-linear differential equations that specify their behavior. However, understanding the nature of these dynamics is not an easy problem [37]. Still, if the research community can demonstrate an ability to formally verify this class of neural networks, they can be used in numerous application domains beyond system identification and fixed-point learning as presented in this paper. Furthermore, CTRNNs have an impressive ability to capture temporal information [1] and can be used to achieve impressive results in problem domains currently dominated by other network architectures. The problems posed in this paper are easily scalable to a given level of complexity by increasing the number of neurons used in implementing the CTRNNs.

There are several interesting possibilities for the extension of the proposed benchmarks and future work. These include:

- Analyzing the dynamics of a system in which a CTRNN is used in conjunction with a controller as described in [19] and [2]
- Studying the effects of using various neuron activation functions on the dynamics of CTRNNs,
- Considering the effects of different neural network training regimes on the reachable set
- Investigating the possibility of linearizing a specific subset of CTRNN models
- Considering multilayer recurrent neural networks.
- and evaluating the possible transformation of other neural network architectures into CTRNNs

References

[1] Design and implementation of multipattern generators in analog vlsi. *IEEE Transactions on Neural Networks*, 17(4):1025–1038, July 2006.

[2] J. C. Atuonwu, Y. Cao, G. P. Rangaiah, and M. O. Tade. Nonlinear model predictive control of a multistage evaporator system using recurrent neural networks. In *2009 4th IEEE Conference on Industrial Electronics and Applications*, pages 1662–1667, May 2009.

[3] I.A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000. Neural Computting in Micrbiology.

[4] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. *CoRR*, abs/1605.07262, 2016.

[5] Randall D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[6] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Piecewise linear neural network verification: A comparative study. *CoRR*, abs/1711.00455, 2017.

[7] Xin Chen, Sriram Sankaranarayanan, and Erika Ábrahám. Under-approximate flowpipes for nonlinear continuous systems. In *Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design*, FMCAD '14, pages 14:59–14:66, Austin, TX, 2014. FMCAD Inc.

[8] Chih-Hong Cheng, Frederik Diehl, Yassine Hamza, Gereon Hinz, Georg Nührenberg, Markus Rickert, Harald Ruess, and Michael Troung-Le. Neural networks for safety-critical applications - challenges, experiments and perspectives. *CoRR*, abs/1709.00911, 2017.

[9] T. W. S. Chow and Xiao-Dong Li. Modeling of continuous time dynamical systems with input by recurrent neural networks. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(4):575–578, Apr 2000.

[10] A. Delgado, C. Kambhampati, and K. Warwick. Dynamic recurrent neural network for system identification and control. *IEE Proceedings - Control Theory and Applications*, 142(4):307–314, Jul 1995.

[11] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. *CoRR*, abs/1705.01320, 2017.

[12] Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. Automatic reachability analysis for nonlinear hybrid models with c2e2. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification*, pages 531–538, Cham, 2016. Springer International Publishing.

[13] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, pages 379–395, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[15] John H. Hubbard. The forced damped pendulum: chaos, complication and control. *Amer. Math. Monthly*, 106:741–758, 1999.

[16] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop. Neural networks for control systems-a survey. *Automatica*, 28(6):1083 – 1112, 1992.

[17] Ken ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801 – 806, 1993.

[18] Eduardo Izquierdo-Torres. On the evolution of continuous time recurrent neural networks with neutrality. 2004.

[19] O. De Jesus, A. Pukrittayakamee, and M. T. Hagan. A comparison of neural network control algorithms. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 1, pages 521–526 vol.1, 2001.

[20] C. Kambhampati, F. Garces, and K. Warwick. Approximation of non-autonomous dynamic systems by continuous time recurrent neural networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, pages 64–69 vol.1, 2000.

[21] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR*, abs/1702.01135, 2017.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[23] Xiao-Dong Li, J. K. L. Ho, and T. W. S. Chow. Approximation of dynamical time-variant systems by continuous-time recurrent neural networks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 52(10):656–660, Oct 2005.

[24] Thomas Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6:e20899, feb 2017.

[25] C. G. Miguel, C. F. d. Silva, and M. L. Netto. Structural and parametric evolution of continuous-time recurrent neural networks. In *2008 10th Brazilian Symposium on Neural Networks*, pages 177–182, Oct 2008.

[26] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[27] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, Sep 1995.

[28] Luca Pulina and Armando Tacchella. Challenging smt solvers to verify neural networks. *AI Commun.*, 25(2):117–135, April 2012.

[29] Karsten Scheibler, Leonore Winterer, Ralf Wimmer, and Bernd Becker. Towards verification of artificial neural networks. In *MBMV*, 2015.

[30] R.K. Al Seyab and Y. Cao. Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. *Journal of Process Control*, 18(6):568 – 581, 2008.

[31] Andrew Sogokon, Khalil Ghorbal, and Taylor T. Johnson. Non-linear continuous systems for safety verification (benchmark proposal). In *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, April 2016.

[32] Andrew Sogokon, Khalil Ghorbal, and Taylor T. Johnson. Non-linear Continuous Systems for Safety Verification (Benchmark Proposal). In Goran Frehse and Matthias Althoff, editors, *ARCH@CPSWeek 2016, 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems, Vienna, Austria*, volume 43 of *EPiC Series in Computing*, pages 42–51, Vienna, Austria, 2016. EasyChair.

[33] Sriram Sanakaranarayanan Ashish Tiwari Souradeep Dutta, Susmit Jha. Output range analysis for deep feed-forward neural networks. 2018. http://www.cs.colorado.edu/~srirams/papers/output_range_analysis_NFM_2018.pdf.

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[35] H. Tang, B. H. Tan, and R. Yan. Robot-to-human handover with obstacle avoidance via continuous time recurrent neural network. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1204–1211, July 2016.

[36] Robert L. V. Taylor. Attractors: Nonstrange to chaotic. 2011.

[37] Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson. Benchmark: A nonlinear reachability analysis test set from numerical analysis. In *ARCH@CPSWeek*, 2015.

[38] Adam P. Trischler and Gabriele M. T. D'Eleuterio. Synthesis of recurrent neural networks for dynamical system simulation. *CoRR*, abs/1512.05702, 2015.

[39] Yong You and Michael Nikolaou. Dynamic process modeling with recurrent neural networks. *AIChE Journal*, 39(10):1654–1667, 1993.

[40] Zhibin Yu, Dennis S. Moirangthem, and Minho Lee. Continuous timescale long-short term memory neural network for human intent understanding. *Frontiers in Neurorobotics*, 11:42, 2017.