HyRNN: Hybrid Recurrent Neural Networks for Approximating Hybrid Dynamical Systems

Ricardo G. Sanfelice[1]

[1]Department of Electrical and Computer Engineering
University of California, Santa Cruz
1156 High Street, Santa Cruz, CA 95060 USA
ricardo@ucsc.edu

Abstract

For a class of hybrid dynamical systems, we show that a recurrent neural network with hybrid dynamics, which we refer to as a hybrid dynamic recurrent neural network (HyRNN), can be constructed to approximate solutions to hybrid systems over bounded (hybrid) time horizons. Specifically, given a desired precision level, we show that a hybrid system with dynamics resembling those of recurrent neural networks for continuous-time and discrete-time systems can be designed so that, for each bounded hybrid time horizon, its solutions are close to the solutions to the given hybrid system. Through the use of universal approximation theorems, we show that the approximation result holds for traditional smooth activation functions, such as sigmoid and arctan, and that extensions to ReLU functions are possible, and characterize the complexity of the proposed HyRNN.

1 Introduction

Neural networks are known to be universal approximators of regular enough functions on compact sets. Approximation results date back to the late 80s/early 90s, though one could argue that the seminal result in (Kolmogorov 1957) establishing that a continuous multivariable function can be approximated by sums and compositions of univariate continuous functions is a precursor to neural network approximation theory. The pioneering result in (Cybenko 1989) establishes that a neural network with a single hidden layer and a sigmoid activation function can approximate any continuous function on a compact set. Around the same time, (Hornik, Stinchcombe, and White 1989) showed that multilayer networks using continuous activation functions are universal approximators of regular enough functions. In (Mhaskar 1996), the complexity of the approximating network is characterized mathematically, launching a quest for complexity optimization to reduce the effect of the curse of dimensionality. These results have been the cornerstone of machine learning, as they provide theoretical guarantees of the approximating capabilities of neural networks. Since then, the focus has been on networks with more than one hidden layer, namely, deep neural networks, with recent advances summarized in (Poggio et al. 2017; Liang and Srikant 2017; Liao 2019; Poggio et al. 2020). Approximation using deep neural networks has led to important in training for reinforcement learning; see, e.g., (Xu et al. 2020; Shen and Yang 2021).

Dynamic recurrent neural networks exploit neural networks to approximate the evolution of the state of a dynamical system. Initial results about the approximation of continuous-time and discrete-time systems by such networks appeared in the literature in the early/mid 90s. For the continuous-time case, (Funahashi and Nakamura 1993) shows that a continuous-time recurrent neural network structure approximates solutions to a nonlinear differential equation using sigmoid functions. Inspired by its approximation properties, (Beer 1995) studies the behavior that can emerge from such networks, focusing on bifurcations, while (Delgado, Verdegay, and Vila 2005) shows that, under appropriate assumptions, such a network is a universal approximator of a nonlinear system. Results are available in the literature for the approximation of the evolution of discrete-time systems. The result in (Jin, Nikiforuk, and Gupta 1995) uses similar techniques as those in (Funahashi and Nakamura 1993) to establish an approximation result for sigmoidal activation functions.

Motivated by the approximating capabilities of dynamic recurrent networks, the goal of this paper is to contribute to neural network approximation theory by introducing a recurrent neural network with hybrid dynamics for the approximation of solutions to a class of hybrid dynamical systems. In this paper, hybrid dynamical systems are modeled by hybrid equations, which are given by the combination of continuous and discrete dynamics (Goebel, Sanfelice, and Teel 2012; Sanfelice 2021), precisely, given by

$$\mathcal{H}: \begin{cases} \dot{x} = f(x) & x \in C \\ x^+ = g(x) & x \in D \end{cases} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system, $C \subset \mathbb{R}^n$ is the *flow set*, which defines the set of points where the state x can evolve continuously, $f : C \to \mathbb{R}^n$ is the *flow map*, which describes the continuous evolution (or flow) of x when it belongs to C, $D \subset \mathbb{R}^n$ is the *jump set*, which is the set of points where the state x can jump, and $g : D \to \mathbb{R}^n$ is the *jump map*, describing the discrete evolution (or jumps) of x when it belongs to D. For this class of systems, which is further described in Section 2, we introduce in Section 3 a hybrid dynamic recurrent neural network (HyRNN) that

flows and jumps in synchrony with $\mathcal{H}$ and generates solutions that approximate the solutions to $\mathcal{H}$ over bounded hybrid time horizons with arbitrary precision. To establish this result, we relate the solutions to two hybrid systems with "approximate" flow and jump maps. We use this relationship in the proof of the approximation result for HyRNNs, which exploits universal approximation theorems for shallow networks to jointly approximate both maps, rather than approximating them separately. The universal approximation theorems lead to shallow or deep networks, depending on the regularity of the given flow and jump map of $\mathcal{H}$ and the activation functions. Basic properties of HyRNN are also presented in Section 3, along with the approximation results. Due to space restrictions, proofs are not included but will be published elsewhere.

Notation: The real, nonnegative, and natural numbers are denoted as $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{N}$, respectively. The matrix $I_{n\times n}$ denotes the $n \times n$ identity matrix and $O_{n\times n}$ denotes the $n \times n$ zero matrix. The Euclidean norm of vectors and the induced matrix norm are denoted by $|\cdot|$. Given vectors x and y, we write $[x^\top y^\top]$ equivalently as (x, y). Given a set S, ∂S denotes its boundary and $\overline{S}$ its closure. Given a function f, the domain of f, $\operatorname{dom} f$, is the set where f is defined, and $\operatorname{rge} f$ is the set of values it attains.

2 Preliminaries

Universal Approximation of Functions

The problem of approximating a given function $\ell : \mathbb{R}^n \to \mathbb{R}^m$ over a compact set $K \subset \mathbb{R}^n$ has been thoroughly studied in the machine learning literature; see, e.g., (Cybenko 1989; Hornik, Stinchcombe, and White 1989; Mhaskar 1996; Poggio et al. 2017; Liao 2019; Poggio et al. 2020; Kidger and Lyons 2020). To approximate ℓ, functions of the form

$$\hat{\ell}(x) := \Gamma\sigma\left(Wx + \theta\right) \qquad \forall x \in \mathbb{R}^n \tag{2}$$

where $\Gamma \in \mathbb{R}^{m\times N}$, $W \in \mathbb{R}^{N\times n}$, and $\theta \in \mathbb{R}^N$ are the parameters to be trained for $\hat{\ell}$ to approximate ℓ on K, and $\sigma : \mathbb{R}^N \to \mathbb{R}^N$ is a given function, called the *activation function*. Precisely, given a desired precision $\varepsilon > 0$, parameters should be tuned so that the error between ℓ and $\hat{\ell}$ is no larger than ε. A function $\hat{\ell}$ with the structure above has only one hidden layer and is known as a *shallow neural network with N units*.

The following fundamental approximation theorem using a continuous nonpolynomial activation functions σ (e.g., sigmoids, tanh, and ReLUs) is well known (Pinkus 1999) (cf. (Funahashi and Nakamura 1993; Cybenko 1989; Hornik, Stinchcombe, and White 1989)).

Theorem 2.1 *(universal approximation theorem with continuous activation functions) Given a compact set $K \subset \mathbb{R}^n$ and a continuous function $\ell : \mathbb{R}^n \to \mathbb{R}^m$, for each $\varepsilon > 0$ there exist $N \in \mathbb{N}$, $\Gamma \in \mathbb{R}^{m\times N}$, $W \in \mathbb{R}^{N\times n}$, and $\theta \in \mathbb{R}^N$ such that $\hat{\ell}$ defined in* (2) *with $\sigma : \mathbb{R}^N \to \mathbb{R}^N$ continuous and nonpolynomial satisfies*

$$|\ell(x) - \hat{\ell}(x)| \leq \varepsilon \qquad \forall x \in K \tag{3}$$

Among the many variations and extensions available in the literature of this approximation result, the one in (Mhaskar 1996, Theorem 2.1) is particularly useful as it allows for any activation function that is smooth, in the sense that it can be differentiated infinitely many times, and, furthermore, also characterizes the complexity of the neural network. Networks that have more than one hidden layer, known as *deep neural networks*, have the potential of reducing the order of the network approximating a given function. The result in (Poggio et al. 2017, Theorem 2)—see also the references therein—formally characterizes the improvement in complexity provided by deep neural networks, at the price of additional structure on the function to approximate. Specifically, the result is for compositional functions $\ell : \mathbb{R}^n \to \mathbb{R}^m$ with a binary tree architecture and constituent functions h that are functions of two variables with r continuous derivatives. Alternatively, the universal approximation result in (Kidger and Lyons 2020), which requires weaker assumptions compared to those mentioned above.

Dynamic Recurrent Neural Networks

Dynamic recurrent neural networks are capable of approximating the solutions to dynamical systems by including internal loops in the paths from inputs to outputs among the units. The literature is rich on the type of such networks, including analysis of their features and shortcomings (Han et al. 2021). A basic model of a dynamic neural network with M units for the approximation of the solutions to a continuous-time system is given by

$$\dot{\mu}_i = -\tau_i\mu_i + \sum_{k=1}^{M} w_{ik}\sigma(\mu_k)$$

for all $i \in \{1, 2, \ldots, M\}$, where, for the i-th unit, μ_i is its internal state, τ_i the time constant, w_{ik} are the weights, and $\sigma(\mu_i)$ is the output of the i-th unit defined by the activation function σ; see, e.g., (Funahashi and Nakamura 1993; Beer 1995). Using the same time constant τ for each unit, the model of the network is given by the nonlinear system

$$\dot{\mu} = -\tau\mu + W\sigma(\mu) \tag{4}$$

A similar model is typically used in the literature for the approximation of solutions to discrete-time systems; see (Jin, Nikiforuk, and Gupta 1995).

In recent years, new dynamic recurrent neural network structures have been proposed in the literature, primarily motivated by the need for improvement in training performance and complexity. The Neural Nonlinear ARX (NNARX) structure (Bonassi, Farina, and Scattolini 2021) is suitable for training that aims at minimizing simulation error. The Echo State Networks (ESN) (Jaeger 2001) can be trained by solving a least squares problem, opening the opportunity to use powerful least square methods already available in the literature. Recently, gated dynamic recurrent neural networks (Hochreiter and Schmidhuber 1997), such as Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs), have been found to be efficient for learning in dynamical systems. However, to the best of the author's knowledge, these emerging dynamic networks have

not been formally shown to approximate solutions to a given system with arbitrary precision. On the other hand, the basic structure in (4) provides such useful approximation; see (Funahashi and Nakamura 1993, Theorem 1).

Hybrid Dynamical Systems

In this paper, we consider hybrid dynamical systems given by as in (1). The data of $\mathcal{H}$ is explicitly denoted as $\mathcal{H} = (C, f, D, g)$. Solutions to $\mathcal{H}$ are defined on hybrid time domains. We introduce these notions in detail.

Definition 2.2 (hybrid time domain) *A* compact hybrid time domain *is a subset of* $\mathbb{R}_{\geq 0} \times \mathbb{N}$ *of the form*

$$\bigcup_{j=0}^{J-1} ([t_j, t_{j+1}], j)$$

for a sequence

$$0 = t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_J$$

with J finite. A set $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ *is a* hybrid time domain *if it is the union of a nondecreasing sequence of compact hybrid time domains, namely, E is the union of compact hybrid time domains* E_j *with the property that*

$$E_0 \subset E_1 \subset E_2 \subset \ldots \subset E_j \ldots$$

A hybrid time horizon is a bounded window of hybrid time of the form $[0, T] \times \{0, 1, \ldots, J\}$.

Solutions to hybrid systems are given by hybrid arcs, which are functions defined on hybrid time domains.

Definition 2.3 (hybrid arc) *A* hybrid arc *is a function* $x : \operatorname{dom} x \to \mathbb{R}^n$ *defined on a hybrid time domain* $\operatorname{dom} x$ *that, for each j,* $t \mapsto x(t, j)$ *is locally absolutely continuous in t.*

We are ready to formally define the notion of solution to the hybrid dynamical system $\mathcal{H}$.

Definition 2.4 (solution to $\mathcal{H}$) *A hybrid arc* $x : \operatorname{dom} x \to \mathbb{R}^n$ *is a* solution *to* $\mathcal{H}$ *if* $x(0, 0) \in \overline{C} \cup D$ *and*

- *For each interval* $I^j := \{t : (t, j) \in \operatorname{dom} x\}$ *with nonempty interior,*

$$\dot{x}(t, j) = f(x(t, j))$$

for almost all $t \in I^j$*, and*

$$x(t, j) \in C$$

for all $t \in \operatorname{int} I^j$*;*

- *At each* $(t, j) \in \operatorname{dom} x$ *such that* $(t, j + 1) \in \operatorname{dom} x$*,*

$$x(t, j) \in D$$

and

$$x(t, j + 1) = g(x(t, j))$$

A solution x to $\mathcal{H}$ is said to be *nontrivial* if its domain contains at least two points, *maximal* if it cannot be further extended, and *complete* if its domain is unbounded. It is shown in (Goebel, Sanfelice, and Teel 2012, Chapter 5) that $\mathcal{H}$ is well-posed if its data satisfies the hybrid basic conditions; namely, the sets C and D are closed, and the maps f and g are continuous. Well-posedness of a hybrid system enables to assert, among other things, that the set of solutions to the system has good structural properties and that asymptotically stable compact sets are robust to small perturbations. Next, we illustrate the framework in a canonical system, a ball bouncing on the ground. For more details, the reader is referred to (Goebel, Sanfelice, and Teel 2012) and (Sanfelice 2021).

A Hybrid Model of a Bouncing Ball

We consider a bouncing ball whose continuous state is

$$x = \begin{bmatrix} p \\ v \end{bmatrix},$$

where p denotes the vertical position (height) and v the vertical velocity. Between impacts the flow dynamics are

$$\dot{p} = v, \qquad \dot{v} = -\gamma,$$

where $\gamma > 0$ is the gravitational constant. When the ball reaches the ground, i.e., when $p = 0$ and $v \leq 0$, a jump occurs and the velocity is reset according to the restitution coefficient $e \in [0, 1]$, as follows:

$$p^+ = 0, \qquad v^+ = -ev.$$

In practice, the physical parameters γ and e are uncertain. Following the hybrid systems framework in (1), the state of $\mathcal{H}$ is given by $x := (p, v) \in \mathbb{R}^2$, the flow and jump sets by

$$C := \{x = (p, v) \in \mathbb{R}^2 : \ p \geq 0\},$$
$$D := \{x = (p, v) \in \mathbb{R}^2 : \ p = 0, \ v \leq 0\},$$

the flow map as

$$f(x) := \begin{bmatrix} v \\ -\gamma \end{bmatrix} \qquad \forall x \in C,$$

and the jump map as

$$g(x) := \begin{bmatrix} 0 \\ -ev \end{bmatrix} \qquad \forall x \in D.$$

The right-hand side of the flow dynamics is globally Lipschitz in x (indeed affine) and the jump map is continuous. Moreover, the flow and jump sets are closed. Then, the data of the hybrid system model satisfies the hybrid basic conditions, making it a well-posed system in the sense of (Goebel, Sanfelice, and Teel 2012; Sanfelice 2021). It can be shown using (Sanfelice 2021, Proposition 2.34) that, for any initial condition $x(0, 0) = (p(0, 0), v(0, 0))$ with $p(0, 0) \geq 0$ there exists a nontrivial solution and that solutions are unique. For $e \in (0, 1)$, the system exhibits *Zeno* behavior: successive impacts occur with geometrically decreasing bounce heights and the sequence of jump times accumulates in finite time. Thus, solutions that start above the ground experience infinitely many jumps in a finite interval of continuous time. Such solutions are maximal and complete.

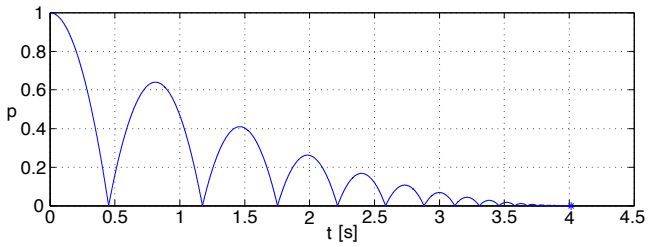

Figure 1: Bouncing ball height as a function of continuous time for restitution coefficient $e = 0.8$, gravitational constant $\gamma = 9.81\ \mathrm{m/s^2}$, initial condition $p(0,0) = 1\,\mathrm{m}$, $v(0,0) = 0$. The plot displays several impacts; after finite time ($\approx 4.1\mathrm{s}$) the impacts accumulate (Zeno).

3 Hybrid Dynamic Recurrent Neural Networks (HyRNNs)

We propose a dynamic recurrent neural network structure as approximator of solutions to $\mathcal{H}$. The proposed network has a hybrid dynamical system structure, with flow and jump dynamics defined as follows: denoting the number of units by $N \in \mathbb{N} \setminus \{0\}$ and its state by $\eta \in \mathbb{R}^N$,

- During flow, its state evolves according to
$$\dot{\eta} = -T_f \eta + W_f \sigma_f(\eta) \tag{5}$$
when $\mathcal{H}$ flows, where $T_f \in \mathbb{R}^N \times \mathbb{R}^N$ is a diagonal matrix representing the time constants of the neurons, $\sigma_f : \mathbb{R}^N \mapsto \mathbb{R}^N$ is the activation function, and $W_f \in \mathbb{R}^N \times \mathbb{R}^N$ is the recurrent weight matrix defining connections between neurons.
- At jumps of $\mathcal{H}$, its state is updated according to
$$\eta^+ = -T_g \eta + W_g \sigma_g(\eta) \tag{6}$$
where, similar to the dynamics during flow, $T_g \in \mathbb{R}^N \times \mathbb{R}^N$ is also a diagonal matrix controlling the state decaying of the neurons, $\sigma_g : \mathbb{R}^N \mapsto \mathbb{R}^N$ is the activation function, and $W_g \in \mathbb{R}^N \times \mathbb{R}^N$ is a matrix defining connections within the units.
- Its output is given by $\zeta = \varrho(\eta)$.

The proposed approximator of $\mathcal{H}$ is denoted $\mathcal{H}_a$ and has hybrid dynamics given by

$$\mathcal{H}_a : \begin{cases} \dot{\eta} = -T_f \eta + W_f \sigma_f(\eta) & \text{when } \mathcal{H} \text{ flows} \\ \eta^+ = -T_g \eta + W_g \sigma_g(\eta) & \text{when } \mathcal{H} \text{ jumps} \\ \zeta = \varrho(\eta) \end{cases} \tag{7}$$

where $\eta \in \mathbb{R}^N$ is the state and $\zeta \in \mathbb{R}^n$ is the output. Due to its hybrid dynamics resembling those of continuous-time and discrete-time recurrent neural networks, we refer to $\mathcal{H}_a$ as a *hybrid dynamic recurrent neural network* (HyRNN).

From its construction, flow and jumps of $\mathcal{H}_a$ are coordinated with those of $\mathcal{H}$. Such coupling prevents hybrid peaking behavior (Sanfelice et al. 2013). Similar to approximators for continuous-time and discrete-time systems, it leads to an interconnection structure where both $\mathcal{H}$ and $\mathcal{H}_a$ evolve concurrently. This interconnection is denoted $\mathcal{H}_{\text{int}}$ and is given by

$$\mathcal{H}_{\text{int}} : \begin{cases} \left.\begin{aligned} \dot{x} &= f(x) \\ \dot{\eta} &= -T_f \eta + W_f \sigma_f(\eta) \end{aligned}\right\} & x \in C \\ \left.\begin{aligned} x^+ &= g(x) \\ \eta^+ &= -T_g \eta + W_g \sigma_g(\eta) \end{aligned}\right\} & x \in D \\ \zeta = \varrho(\eta) \end{cases} \tag{8}$$

Given $\mathcal{H} = (C, f, D, g)$, $N \in \mathbb{N} \setminus \{0\}$, and activation functions σ_f and σ_g, the objective is to design the data $(T_f, W_f, T_g, W_g, \varrho)$ of $\mathcal{H}_a$ so that, in the spirit of (Funahashi and Nakamura 1993, Theorem 1), for given desired precision, solutions to $\mathcal{H}_a$ approximate those of $\mathcal{H}$ with such precision. To this end, as it is common practice when approximating continuous-time or discrete-time systems using recurrent neural networks (see Section 2), a universal approximation theorem will be employed to show that $\mathcal{H}_a$ approximates $\mathcal{H}$. For simplicity, we consider the case of shallow networks, but the structure of HyRNN allows for the use of deep networks, as those are given by compositions of linear combinations of activation functions. Hence, we will require the flow map f and jump map g to satisfy the conditions in the approximation theorem in Theorem 2.1.[1]

Assumption 3.1 *Given scalar activation functions $\bar{\sigma}_f, \bar{\sigma}_g : \mathbb{R} \to \mathbb{R}$, and the data (C, f, D, g) of $\mathcal{H}$, the sets C and D are compact, f and g are continuous on C and D, respectively, and $\bar{\sigma}_f$ and $\bar{\sigma}_g$ are continuous nonpolynomial functions such that $\bar{\sigma}_\star(s) = 0$ implies $s = 0$ for each $\star \in \{f, g\}$.*

The scalar activation functions $\bar{\sigma}_f$ and $\bar{\sigma}_g$ define the vector activation functions σ_f and σ_g – see footnote 1. Note that sigmoids, tanh, and leaky ReLUs satisfy Assumption 3.1.

Basic Properties

Solutions to $\mathcal{H}$ are defined over hybrid time domains, which might be bounded, compact, and even unbounded; see Section 2. For the interconnection $\mathcal{H}_{\text{int}}$, existence of solutions depends on the data of $\mathcal{H}$ and $\mathcal{H}_a$. The following result characterizes its solutions. It assumes the activation function is Lipschitz, which a property that the vast majority of activation functions satisfies (e.g., leaky ReLU, SoftPlus, tanh, sigmoid, arctan, among others).

Proposition 3.2 *(existence of solutions) Let a compact set K and $\mathcal{H}$ as in (1) be given. Suppose that $\sigma_f : \mathbb{R}^N \to \mathbb{R}^N$ is Lipschitz. Then, for each solution $x : \operatorname{dom} x \to \mathbb{R}^n$ to $\mathcal{H}$ such that*
$$x(t, j) \in K$$
for all $(t, j) \in \operatorname{dom} x$, and each $\eta_\circ \in \mathbb{R}^N$, there exists a hybrid arc $\eta : \operatorname{dom} \eta \to \mathbb{R}^N$ such that $\operatorname{dom} \eta = \operatorname{dom} x$ and (x, η) is a solution to $\mathcal{H}_{\text{int}}$ with $\eta(0, 0) = \eta_\circ$.

As established in (Goebel, Sanfelice, and Teel 2012), well-posedness of a hybrid system allows to relate nominal solutions to its perturbations over compact hybrid time domains, and also certify that asymptotic stability of a compact

[1] For the deep network case, one can use (Poggio et al. 2017, Theorem 2); see also (Liang and Srikant 2017).

set is robust to small perturbations. It can be shown that $\mathcal{H}_{\text{int}}$ is well-posed when $\mathcal{H}$ satisfies the hybrid basic conditions and the activation functions are continuous.

Approximation Properties of HyRNN

In this section, we show that, under appropriate conditions, $\mathcal{H}_a$ in (7) approximates the solutions to $\mathcal{H}$. To that end, first, we present a result that, over compact hybrid time domains, relates solutions to two hybrid systems with approximately the same flow and jump map.

Proposition 3.3 *(hybrid tube bound over compact hybrid time domains) Given a hybrid system $\mathcal{H} = (C, f, D, g)$ as in* (1) *with C and D compact, suppose*

1. *f is Lipschitz on C with Lipschitz constant L_f;*
2. *g is Lipschitz on D with Lipschitz constant L_g;*

and that there exist $\varepsilon_f > 0$, $\varepsilon_g > 0$, a locally Lipschitz function $\hat{f}$ and a function $\hat{g}$ such that

$$|f(x) - \hat{f}(x)| \leq \varepsilon_f \quad \forall x \in C + \widetilde{\varepsilon}\mathbb{B} \tag{9}$$

$$|g(x) - \hat{g}(x)| \leq \varepsilon_g \quad \forall x \in D + \widetilde{\varepsilon}\mathbb{B} \tag{10}$$

where $\widetilde{\varepsilon} := \max\{\varepsilon_f, \varepsilon_g\}$. Then, each solution x to

$$\mathcal{H} : \begin{cases} \dot{x} = f(x) & x \in C \\ x^+ = g(x) & x \in D \end{cases}$$

each solution $\hat{x}$ to

$$\hat{\mathcal{H}} : \begin{cases} \dot{\hat{x}} = \hat{f}(\hat{x}) & \text{when } \mathcal{H} \text{ flows} \\ \hat{x}^+ = \hat{g}(\hat{x}) & \text{when } \mathcal{H} \text{ jumps} \end{cases} \tag{11}$$

such that[2] $\operatorname{dom} x = \operatorname{dom} \hat{x}$*, and each $(T, J) \in \operatorname{dom} x$ satisfy*[3]

$$|x(t,j) - \hat{x}(t,j)| \leq \left. \underset{j=0}{\overset{J}{\circ}} \rho_j \right|_{r=|x(0,0)-\hat{x}(0,0)|} \tag{12}$$

for all $(t, j) \in \operatorname{dom} x$ such that $t + j \leq T + J$, where, for each $j \in \{0, 1, \ldots, J\}$,

$$\rho_j(r) := \rho(r, \tilde{I}^j)$$

and, with $\tilde{I}^j = \{t \ : \ (t, j) \in \operatorname{dom} x, t + j \leq T + J\}$,

$$\begin{aligned} \rho(r, \tilde{I}^j) &:= \rho_f(\rho_g(r, \tilde{I}^j), \tilde{I}^j) \\ \rho_f(r, \tilde{I}^j) &:= r b_f(\tilde{I}^j) + a_f(b_f(\tilde{I}^j) - 1) \\ \rho_g(r, \tilde{I}^j) &:= (a_g r + \varepsilon_g)\, b_g(\tilde{I}^j) + r(1 - b_g(\tilde{I}^j)) \\ a_f &:= \frac{\varepsilon_f}{L_f}, \qquad a_g := L_g \\ b_f([s_0, s_1]) &:= \exp(L_f(s_1 - s_0)) \end{aligned}$$

and $b_g([s_0, s_1])$ is equal to one if s_0 is a jump time of x and $\hat{x}$ and zero otherwise. Furthermore, when the initial condition for x and for $\hat{x}$ are the same, the bound is linear in $\widetilde{\varepsilon}$, namely, for each solution x to $\mathcal{H}$ in (1)*, each solution $\hat{x}$ to $\hat{\mathcal{H}}$ in* (11) *with $\hat{x}(0,0) = x(0,0)$, and each $(T, J) \in \operatorname{dom} x (= \operatorname{dom} \hat{x})$, x and $\hat{x}$ satisfy*

$$|x(t,j) - \hat{x}(t,j)| \leq \alpha \widetilde{\varepsilon}$$

for all $(t, j) \in \operatorname{dom} x$ such that $t + j \leq T + J$, where α is independent of $\widetilde{\varepsilon}$.

Now, we are ready to introduce our main result showing that, under the stated assumptions, there exists a HyRNN as in (7) that, for given precision, approximates the solutions to the hybrid system $\mathcal{H}$ in (1) over bounded hybrid time horizons.

Theorem 3.4 *(approximation by HyRNN) Given a hybrid system $\mathcal{H} = (C, f, D, g)$ as in* (1)*, $\varepsilon > 0$, and scalar activation functions $\bar{\sigma}_f, \bar{\sigma}_g : \mathbb{R} \to \mathbb{R}$, suppose*

- *Assumption 3.1 holds;*
- *f is Lipschitz on $C + \varepsilon\mathbb{B}$ with Lipschitz constant L_f;*
- *g is Lipschitz on $D + \varepsilon\mathbb{B}$ with Lipschitz constant L_g.*

Then, there exist

- *an integer N;*
- *matrices T_f, W_f, T_g, $W_g \in \mathbb{R}^N \times \mathbb{R}^N$;*

such that, along with $\sigma_f : \mathbb{R}^N \to \mathbb{R}^N$ and $\sigma_g : \mathbb{R}^N \to \mathbb{R}^N$ given as

$$\sigma_f := (\bar{\sigma}_f, \bar{\sigma}_f, \ldots, \bar{\sigma}_f)$$

and

$$\sigma_g := (\bar{\sigma}_g, \bar{\sigma}_g, \ldots, \bar{\sigma}_g)$$

defining the data of the hybrid dynamic recurrent neural network $\mathcal{H}_a$ in (7) *the following holds: for each solution x to $\mathcal{H}$ in* (1) *and each $(T, J) \in \operatorname{dom} x$, there exists a solution η to $\mathcal{H}_a$ with output ζ such that*

$$|x(t,j) - \zeta(t,j)| \leq \varepsilon \tag{13}$$

for all $(t, j) \in \operatorname{dom} x \cap ([0, T] \times \{0, 1, \ldots, J\})$.

Proof sketch: Let $\varepsilon > 0$, a solution x to $\mathcal{H}$, and $(T, J) \in \operatorname{dom} x$ be given. By a universal approximation theorem (see Section 2) with

$$K = (C \cup D) + \varepsilon\mathbb{B}$$

there exist $N > 0$, $\widetilde{\Gamma} \in \mathbb{R}^{2n \times 2N}$, $\widetilde{W} \in \mathbb{R}^{2N \times n}$, and $\widetilde{\theta} \in \mathbb{R}^{2N}$, such that, with

$$\widetilde{\sigma} := (\sigma_f, \sigma_g) : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$$

the map

$$\begin{bmatrix} \widetilde{f}(x) \\ \widetilde{g}(x) \end{bmatrix} := \widetilde{\Gamma}\widetilde{\sigma}(\widetilde{W}x + \widetilde{\theta}) \qquad \forall x \in \mathbb{R}^n$$

satisfies

$$|\widetilde{f}(x) - f(x)| \leq \frac{\bar{\varepsilon}}{2} \qquad \forall x \in C + \varepsilon\mathbb{B} \tag{14}$$

$$|\widetilde{g}(x) - g(x)| \leq \frac{\bar{\varepsilon}}{2} \qquad \forall x \in D + \varepsilon\mathbb{B} \tag{15}$$

[2] Since $\hat{f}$ is Lipschitz, and the interval of flow and jump times of solutions to $\hat{\mathcal{H}}$ coincide with those of solutions to $\mathcal{H}$, the existence of a solution $\hat{x}$ to $\hat{\mathcal{H}}$ follows from the same steps in the proof of Proposition 3.2.

[3] In (12), $\circ_{j=0}^{J} \rho_j$ means the composition $\rho_J \circ \rho_{J-1} \circ \ldots \rho_1 \circ \rho_0$.

1. Inspired by the construction in the proof of (Funahashi and Nakamura 1993, Theorem 1), we define the auxiliary hybrid system $\hat{\mathcal{H}}$ defined as in (11) with maps $\hat{f}$ and $\hat{g}$ given by

$$\hat{f}(x, \widetilde{\theta}) := -\tau_f x + E_f \widetilde{\Gamma} \widetilde{\sigma}(\widetilde{W} x + \widetilde{\theta}) \qquad \forall x \in \mathbb{R}^n \quad (16)$$

$$\hat{g}(x, \widetilde{\theta}) := -\tau_g x + E_g \widetilde{\Gamma} \widetilde{\sigma}(\widetilde{W} x + \widetilde{\theta}) \qquad \forall x \in \mathbb{R}^n \quad (17)$$

where τ_f and τ_g are positive constants that are yet to be tuned and

$$E_f := [I_{n\times n} \quad O_{n\times n}], E_g := [O_{n\times n} \quad I_{n\times n}]$$

where $I_{n\times n}$ is the $n \times n$ identify matrix and $O_{n\times n}$ is the $n \times n$ zero matrix. Then, employing $\hat{\mathcal{H}}$ as an intermediate approximator of the solutions to $\mathcal{H}$, we relate the solutions x to $\mathcal{H}$ and the solutions $\hat{x}$ to $\hat{\mathcal{H}}$ via a bound depending on ε over $[0, T] \times \{0, 1, \ldots, J\}$.

2. We define a hybrid system, denoted $\widetilde{\mathcal{H}}$, with the same structure as $\mathcal{H}_a$ and a state component that: i) approximates θ and ii) generates an output $\widetilde{\eta}$ that approximates solutions $\hat{x}$, also via a bound depending on ε and over $[0, T] \times \{0, 1, \ldots, J\}$.

3. Finally, we define a hybrid system, denoted $\widetilde{\mathcal{H}}_{NN}$, with output $\widetilde{\eta}$ that captures the dynamics of $\widetilde{\mathcal{H}}$ and is of the form of $\mathcal{H}_a$. ■

Approximating the Bouncing Ball

We revisit the hybrid model of the bouncing ball in presented in Section 2, and implement and train a HyRNN approximating its solutions. The HyRNN contains two separate recurrent subnetworks, one for the flow dynamics and one for the jump dynamics. The subnetworks implement corrections and normalization for efficient training.

The implemented training strategy minimizes a combined flow and jump loss. The flow loss is defined using the right-hand side of the flow dynamics of each system, namely,

$$\mathcal{J}_{\text{flow}}(x_{\text{dot,pred}}, x_{\text{dot,true}}) = \frac{1}{M} \sum_{i=1}^{N} \left| x^{(i)}_{\text{dot,pred}} - x^{(i)}_{\text{dot,true}} \right|^2$$

where M is the number of samples in the batch, $x_{\text{dot,pred}}$ denotes the value of the right-hand side associated with the flow map of the HyRNN while $x_{\text{dot,true}}$ is the corresponding value of the flow map of the bouncing ball system. Similarly, the jump loss is defined as

$$\mathcal{J}_{\text{jump}}(x_{\text{jump,pred}}, x_{\text{jump,true}}) = \frac{1}{J} \sum_{j=1}^{J} \left| x^{(j)}_{\text{jump,pred}} - x^{(j)}_{\text{jump,true}} \right|^2$$

where J is the number of jumps. The total loss is then

$$\mathcal{J}_{\text{total}} = \mathcal{J}_{\text{flow}} + \lambda_{\text{jump}} \mathcal{J}_{\text{jump}} \quad (18)$$

with λ_{jump} is a positive tunable constant. Training is conducted using the Adam optimizer with a learning rate of 1×10^{-3} and weight decay 1×10^{-6}. To avoid gradient explosion, gradients are clipped with a maximum norm of 1.0. Using leaky RELU activation functions, the network is trained for s^* epochs with a time step of $s = 0.002\,\text{s}$. Curriculum learning is employed, starting with small maximum initial heights ($0.2\,\text{m}$) and gradually increasing to $2.0\,\text{m}$ over the first 20% of epochs. This gradually exposes the network to more energetic trajectories, stabilizing training. During each training iteration, the flow hidden state is updated continuously at each time step, while the jump hidden state is updated only for those samples where an impact occurs. The flow loss ensures accurate prediction of the continuous dynamics, while the jump loss emphasizes correct handling of jumps.

Training data is collected by randomly sampling initial positions and velocities. For each batch, position p is sampled uniformly between zero and the current maximum height determined by the curriculum, and velocity v is sampled uniformly from $[-1, 1]\,\text{m/s}$. The states are normalized This normalization ensures that all input values are roughly centered and scaled for stable training.

At a high level, the training loop initializes hidden states and normalized inputs for each batch. For each time step, the flow subnetwork predicts a correction x_{corr}, which is added to the baseline derivative to produce $x_{\text{dot,pred}}$. The flow loss is computed as the mean squared error between predicted and true derivatives. When an impact occurs, the jump subnetwork predicts the next state, and a jump loss is computed as the squared error. The total loss accumulates contributions from flow and jump terms over all steps, and backpropagation updates the HyRNN parameters.

Listing 1: HyRNN training pseudocode.

```
for epoch in range(num_epochs):
    x0 = sample_initial_states(
        batch_size)
    h_flow = zeros(flow_hidden_dim)
    h_jump = zeros(jump_hidden_dim)
    loss_epoch = 0

    for t in range(num_steps):
        # Flow dynamics
        x_dot_true = flow_dynamics(x)
        h_dot_flow, x_corr = rnn.
            forward_flow(h_flow)
        x_dot_pred = baseline_flow +
            corr_scale * x_corr
        loss_flow = mse(x_dot_pred,
            x_dot_true)

        # Jump dynamics
        if impact_event:
            h_jump_pred, x_jump_pred =
                rnn.forward_jump(h_jump)
            loss_jump = mse(x_jump_pred,
                x_jump_true)
        else:
            loss_jump = 0

        # Accumulate loss
        loss_epoch += loss_flow +
            jump_weight * loss_jump
```

```
            # Update states
            x = x + dt * x_dot_pred
            h_flow = h_flow + dt *
                h_dot_flow

        # Backpropagation
        optimizer.zero_grad()
        loss_epoch.backward()
        clip_grad_norm(rnn.parameters(),
            1.0)
        optimizer.step()
```

The HyRNN effectively captures both continuous and discrete dynamics of the bouncing ball. The flow subnetwork provides accurate corrections to a baseline linear prediction, while the jump subnetwork ensures correct handling of impacts. Curriculum learning and appropriate loss weighting allow the network to learn complex hybrid behaviors with low trajectory error. After training, the HyRNN is evaluated on long trajectories using RK4 integration for the true dynamics. Figure 2 shows the loss function Solutions for $\gamma = 9.81\ \mathrm{m/s^2}$ and $e = 0.8$ are compared in Figure 3. The solution to the bouncing ball model, which exhibits Zeno, is show in solid (blue) and the solution to its associated HyRNN is show in dashed (orange). The Euclidean error between true and predicted trajectories is shown in Figure 4. These plots show that the error between the solutions is small, as guaranteed by Theorem 2.1.

A complete implementation of the HyRNN training and evaluation framework for the bouncing ball system is available in the public GitHub repository HybridSystemsLab/HyRNN-BouncingBall. The repository contains the full training and plotting scripts, along with supporting modules and documentation. All simulations reported here were executed on a MacBook Pro with an Apple M3 Pro processor, and the full training and evaluation process required approximately 45 minutes.

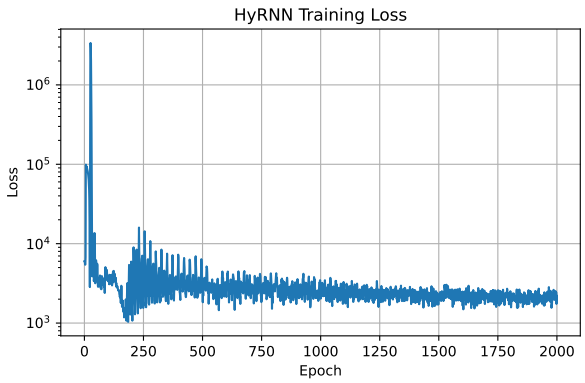

Figure 2: HyRNN training loss over epochs of training in log scale.

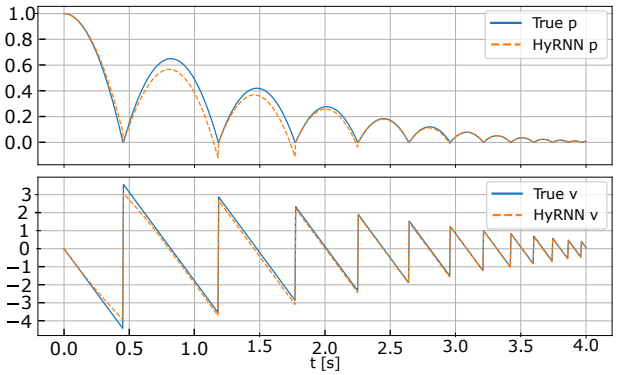

Figure 3: Comparison of solutions to the bouncing ball system (true) and its associated HyRNN over ordinary time.

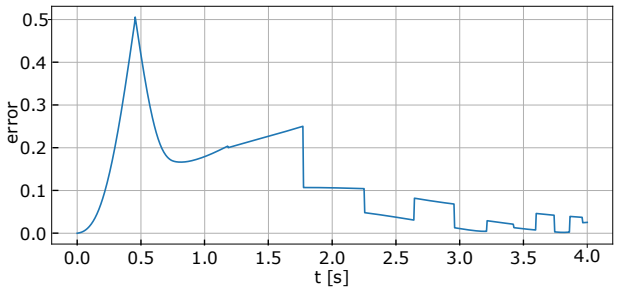

Figure 4: Euclidean norm error between true and HyRNN-predicted trajectories.

4 Conclusion

HyRNN, a dynamic recurrent neural network for approximation of solutions to hybrid dynamical systems is introduced. Under mild conditions known to be required for approximation of dynamical systems using neural networks, HyRNN is able to approximate solutions to a hybrid dynamical system with arbitrary precision. Ongoing work includes formally treating the more tedious case of deep networks and numerical experiments implementing training schemes that efficiently handle the combination of flows and jumps. Future work includes approximating the flow and jump sets, and devising training methods for HyRNN.

Acknowledgments

The author would like to thank Dr. Himadri Basu and Dr. Daniel Ochoa for useful suggestions. Research by partially supported by NSF Grants no. CNS-2039054 and CNS-2111688, by AFOSR Grants nos. FA9550-23-1-0145, FA9550-23-1-0313, and FA9550-23-1-0678, by AFRL Grant nos. FA8651-22-1-0017 and FA8651-23-1-0004, by ARO Grant no. W911NF-20-1-0253, and by DoD Grant no. W911NF-23-1-0158.

References

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

Bonassi, F.; Farina, M.; and Scattolini, R. 2021. Stability of discrete-time feedforward neural networks in NARX configuration. In *19th IFAC Symposium on System Identification (SYSID 2021)*. IFAC.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4): 303–314.

Delgado, M.; Verdegay, J. L.; and Vila, M. A. 2005. Dynamic recurrent neural network for system identification and control. *Neurocomputing*, 70(16-18): 2690–2700.

Funahashi, K.-I.; and Nakamura, Y. 1993. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6): 801–806.

Goebel, R.; Sanfelice, R. G.; and Teel, A. R. 2012. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. New Jersey: Princeton University Press.

Han, Y.; Huang, G.; Song, S.; Yang, L.; Wang, H.; and Wang, Y. 2021. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11): 7436–7456.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5): 359–366.

Jaeger, H. 2001. The echo state approach to analysing and training recurrent neural networks—with an erratum note. Technical Report 148, German National Research Center for Information Technology (GMD), Bonn, Germany.

Jin, L.; Nikiforuk, P. N.; and Gupta, M. M. 1995. Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks. *IEEE Transactions on Automatic Control*, 40(5): 879–884.

Kidger, P.; and Lyons, T. 2020. Universal approximation with deep narrow networks. In *Conference on learning theory*, 2306–2327. PMLR.

Kolmogorov, A. N. 1957. On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, 114: 953–956.

Liang, S.; and Srikant, R. 2017. Why Deep Neural Networks for Function Approximation? *arXiv preprint, https://arxiv.org/abs/1610.04161*.

Liao, Q. 2019. Understanding neural networks from theoretical and biological perspectives. *arXiv preprint, https://arxiv.org/abs/1908.09385*.

Mhaskar, H. 1996. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8(1): 164–177.

Pinkus, A. 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8: 143–195.

Poggio, T.; Mhaskar, H.; Liao, Q.; Miranda, B.; and Rosasco, L. 2020. Theoretical issues in deep networks: Approximation, optimization and generalization. *arXiv preprint, https://arxiv.org/abs/2001.03606*.

Poggio, T.; Mhaskar, H.; Rosasco, L.; Miranda, B.; and Liao, Q. 2017. Why and when can deep–but not shallow–networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5): 503–519.

Sanfelice, R. G. 2021. *Hybrid Feedback Control*. New Jersey: Princeton University Press.

Sanfelice, R. G.; Biemond, J. J. B.; van de Wouw, N.; and Heemels, W. P. M. H. 2013. An Embedding Approach for the Design of State-Feedback Tracking Controllers for References with Jumps. *International Journal of Robust and Nonlinear Control*, 24(11): 1585–1608.

Shen, J.; and Yang, L. F. 2021. Theoretically principled deep RL acceleration via nearest neighbor function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9558–9566.

Xu, S.; Panwar, S. S.; Kodialam, M.; and Lakshman, T. 2020. Deep neural network approximated dynamic programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1684–1691.