2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Two-Level Lattice Neural Network Architectures for Control of Nonlinear Systems

James Ferlez*†, Xiaowu Sun*†, and Yasser Shoukry*

Abstract— **In this paper, we consider the problem of automatically designing a Rectified Linear Unit (ReLU) Neural Network (NN) architecture (number of layers and number of neurons per layer) with the guarantee that it is sufficiently parametrized to control a nonlinear system. Whereas current state-of-the-art techniques are based on hand-picked architectures or heuristic-based search to find such NN architectures, our approach exploits a given model of the system to design an architecture; as a result, we provide a guarantee that the resulting NN architecture is sufficient to implement a controller that satisfies an achievable specification. Our approach exploits two basic ideas. First, we assume that the system can be controlled by a Lipschitz-continuous state-feedback controller that is unknown but whose Lipschitz constant is upper-bounded by a known constant; then using this assumption, we bound the number of affine functions needed to construct a Continuous Piecewise Affine (CPWA) function that can approximate the unknown Lipschitz-continuous controller. Second, we utilize the authors' recent results on the Two-Level Lattice (TLL) NN architecture, a novel NN architecture that was shown to be parameterized directly by the number of affine functions that comprise the CPWA function it realizes. We also evaluate our method by designing a NN architecture to control an inverted pendulum.**

I. Introduction

Multilayer Neural Networks (NN) have shown tremendous success in realizing feedback controllers that can achieve several complex control tasks [1]. Nevertheless, the current state-of-the-art practices for designing these deep NN-based controllers are based on heuristics and hand-picked hyper-parameters (e.g., number of layers, number of neurons per layer, training parameters, training algorithm) without an underlying theory that guides their design. For example, several researchers have studied the problem of Automatic Machine Learning (AutoML) and in particular the problem of hyperparameter (number of layers, number of neurons per layer, and learning algorithm parameters) optimization and tuning for deep NNs (see for example [2], [3], [4], [5], [6] and the references within). These methods perform an iterative and exhaustive search through a manually specified subset of the hyperparameter space; the best hyperparameters are then selected according to some performance metric without any guarantee on the correctness of the chosen architecture.

In this paper, we exhibit a systematic methodology for choosing a NN controller architecture (number of layers and number of neurons per layer) to control a nonlinear system. Specifically, we design an architecture that is guaranteed to correctly control a nonlinear system in the following sense: there exist neuron weights/biases for the designed architecture such that it can meet exactly the same specification as any other continuous, non-NN controller with at most an a priori specified Lipschitz constant. Moreover, provided such a non-NN controller exists (to help ensure well-posedness), the design of our architecture requires only knowledge of a bound on such a controller's Lipschitz constant; the robustness of the specification; and the Lipschitz constants/vector field bound of the nonlinear system. Thus, our approach may be applicable even without perfect knowledge of the underlying system dynamics, albeit at the expense of designing rather larger architectures.

Our approach exploits several insights. First, state-of-the-art NNs use Rectified Linear Units (ReLU), which in turn restricts such NN controllers to implement only Continuous Piecewise Affine (CPWA) functions. As is widely known, a CPWA function is compromised of several affine functions (named local linear functions), which are defined over a set of polytypic regions (called local linear regions). In other words, a ReLU NN—by virtue of its CPWA character—partitions its input space into a set of polytypic regions (named activation regions), and applies a linear controller at each of these regions. Therefore, a NN architecture dictates the number of such activation regions in the corresponding CPWA function that is represented by the trainable parameters in the NN. That is, to design a NN architecture, one needs to perform two steps: (i) compute (or upper bound) the number of activation regions required to implement a controller that satisfies the specifications; and (ii) transform this number of activation regions into a NN architecture that is guaranteed to give rise to this number of activation regions.

To count the number of the required activation regions, we start by assuming the existence of a Lipschitz-continuous, state-feedback controller that can robustly control the nonlinear system to meet given specifications. However, as stated above, we make no further assumptions about this controller except that its Lipschitz constant is upper-bounded by a known constant, K_{cont}. Using this Lipschitz-constant bound – *but no other specific information about the controller* – together with the Lipschitz constants/vector field bound of the system and robustness of the specification, we exhibit an upper-bound for the number of activation regions needed to approximate this controller by a CPWA controller, while still meeting the same specifications in closed loop.

Next, we leverage this bound on activation regions using the authors' recent results on a novel NN architecture, the Two-Level Lattice (TLL) NN architecture [7]. Unlike other NN architectures where the number of activation regions is not explicitly specified, the TLL-NN architecture is explicitly parametrized by the number of activation regions it contains. Thus, we can directly specify a TLL architecture from the aforementioned bound on the number of activation regions. The resulting NN architecture is then guaranteed to be sufficiently parametrized to implement a CPWA function that approximates the unknown Lipschitz-continuous controller in such a way that the specification is still met. This provides a

† Equally contributing first authors.

*Department of Electrical Engineering and Computer Science, University of California, Irvine {jferlez,xiaowus,yshoukry}@uci.edu

This work was partially sponsored by the NSF awards #CNS-2002405 and #CNS-2013824.

systematic approach to designing a NN architecture for such systems.

II. Preliminaries

A. Notation

We denote by $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{R}^+$ the set of natural numbers, the set of real numbers and the set of non-negative real numbers, respectively. For a function $f : A \to B$, let $\text{dom}(f)$ return the domain of f, and let $\text{range}(f)$ return the range of f. For a set $V \subseteq \mathbb{R}^n$, let $\text{int}(V)$ return the interior of V. For $x \in \mathbb{R}^n$, we will denote by $\|x\|$ the infinity norm of x; for $x \in \mathbb{R}^n$ and $\epsilon \geq 0$ we will denote by $B(x;\epsilon)$ the ball of radius ϵ centered at x as specified by $\|\cdot\|$. For $f : \mathbb{R}^n \to \mathbb{R}^m$, $\|f\|_\infty$ will denote the essential supremum norm of f. Finally, given two sets A and B denote by B^A the set of all functions $f : A \to B$.

B. Dynamical Model

In this paper, we will assume an underlying, but not necessarily known, continuous-time nonlinear dynamical system specified by an ordinary differential equation (ODE): that is

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

where the state vector $x(t) \in \mathbb{R}^n$ and the control vector $u(t) \in \mathbb{R}^m$. Formally, we have the following definition:

Definition 1 (Control System)**.** *A **control system** is a tuple $\Sigma = (X, U, \mathcal{U}, f)$ where*

- *$X \subset \mathbb{R}^n$ is the compact subset of the state space;*
- *$U \subset \mathbb{R}^m$ is the compact set of admissible (instantaneous) controls;*
- *$\mathcal{U} \subseteq U^{\mathbb{R}^+}$ is the space of admissible open-loop control functions – i.e. $v \in \mathcal{U}$ is a function $v : \mathbb{R}^+ \to U$; and*
- *$f : \mathbb{R}^n \times U \to \mathbb{R}^n$ is a vector field specifying the time evolution of states according to (1).*

*A control system is said to be (globally) **Lipschitz** if there exists constants K_x and K_u such that for all $x, x' \in \mathbb{R}^n$ and $u, u' \in \mathbb{R}^m$:*

$$\|f(x,u) - f(x',u')\| \leq K_x\|x - x'\| + K_u\|u - u'\|. \tag{2}$$

For a Lipschitz control system, the following vector field bound is well defined:

$$\mathcal{K} \triangleq \max_{x \in X, u \in U} \|f(x,u)\|. \tag{3}$$

In the sequel, we will primarily be concerned with solutions to (1) that result from instantaneous state-feedback controllers, $\Psi : X \to U$. Thus, we use $\zeta_{x_0\Psi}$ to denote the *closed-loop* solution of (1) starting from initial condition x_0 (at time $t = 0$) and using *state-feedback controller* Ψ. We refer to such a $\zeta_{x_0\Psi}$ as a (closed-loop) *trajectory* of its associated control system.

Definition 2 (Closed-loop Trajectory)**.** *Let Σ be a Lipschitz control system, and let $\Psi : \mathbb{R}^n \to U$ be a globally Lipschitz continuous function. A **closed-loop trajectory** of Σ under controller Ψ and starting from $x_0 \in X$ is the function $\zeta_{x_0\Psi} : \mathbb{R}^+ \to X$ that uniquely solves the integral equation:*

$$\zeta_{x_0\Psi}(t) = x_0 + \int_0^t f(\zeta_{x_0\Psi}(\sigma), \Psi(\zeta_{x_0\Psi}(\sigma)))d\sigma. \tag{4}$$

It is well known that such solutions exist and are unique under these assumptions [8].

Definition 3 (Feedback Controllable)**.** *A Lipschitz control system Σ is **feedback controllable** by a Lipschitz controller $\Psi : \mathbb{R}^n \to U$ if the following is satisfied:*

$$\Psi \circ \zeta_{x\Psi} \in \mathcal{U} \qquad \forall x \in X. \tag{5}$$

If Σ is feedback controllable for any such Ψ, then we simply say that it is feedback controllable.

Because we're interested in a compact set of states, X, we consider only feedback controllers whose closed-loop trajectories stay within X.

Definition 4 (Positive Invariance)**.** *A feedback trajectory of a Lipschitz control system, $\zeta_{x_0\Psi}$, is **positively invariant** if $\zeta_{x_0\Psi}(t) \in X$ for all $t \geq 0$. A controller Ψ is positively invariant if $\zeta_{x_0\Psi}$ is positively invariant for all $x_0 \in X$.*

For technical reasons, we will also need the following stronger notion of positive invariance.

Definition 5 (δ,τ Positive Invariance)**.** *Let $\delta, \tau > 0$ and $edge_\delta(X) \triangleq \cup_{x \in X \setminus int(X)}(X \cap B(x;\delta))$. Then a positively invariant controller Ψ is δ,τ **positively invariant** if*

$$\forall x_0 \in edge_\delta(X) \,.\, \zeta_{x_0\Psi}(\tau) \in X \setminus edge_\delta(X) \tag{6}$$

and Ψ is positively invariant with respect to $X \setminus edge_\delta(X)$.

For a δ,τ positively invariant controller, trajectories that start δ-close to the boundary of X end up at least δ-far *away* from that boundary after τ seconds, and remain there forever after.

Finally, borrowing from [9], we define a τ-sampled transition system embedding of a feedback-controlled system.

Definition 6 (τ-sampled Transition System Embedding)**.** *Let $\Sigma = (X, U, \mathcal{U}, f)$ be a feedback controllable Lipschitz control system, and let $\Psi : \mathbb{R}^n \to U$ be a Lipschitz continuous feedback controller. For any $\tau > 0$, the **τ-sampled transition system embedding** of Σ under Ψ is the tuple $S_\tau(\Sigma_\Psi) = (X_\tau, \mathcal{U}_\tau, \xrightarrow[\Sigma_\Psi]{})$ where:*

- *$X_\tau = X$ is the state space;*
- *$\mathcal{U}_\tau = \{(\Psi \circ \zeta_{x_0\Psi})|_{t\in[0,\tau]} : x_0 \in X\}$ is the set of open loop control inputs generated by Ψ-feedback, each restricted to the domain $[0,\tau]$; and*
- *$\xrightarrow[\Sigma_\Psi]{} \subseteq X_\tau \times \mathcal{U}_\tau \times X_\tau$ such that $x \xrightarrow[\Sigma_\Psi]{u} x'$ iff both $u = (\Psi \circ \zeta_{x\Psi})|_{t\in[0,\tau]}$ and $x' = \zeta_{x\Psi}(\tau)$.*

*$S_\tau(\Sigma_\Psi)$ is thus a **metric** transition system [9].*

C. Abstract Disturbance Simulation

In this subsection, we propose a new simulation relation, which we call *abstract disturbance simulation*, as a formal notion of specification satisfaction for metric transition systems. Abstract disturbance simulation is inspired by robust bisimulation [10] and especially disturbance bisimulation [11], but it abstracts those notions away from their definitions in terms of control system embeddings and explicit modeling of disturbance inputs. In this way, it is conceptually similar to the technique used in [9] and [12] to define a quantized abstraction, where deliberate non-determinism is introduced in order to account for input errors. As a prerequisite, we introduce the following definition.

Definition 7 (Perturbed Metric Transition System)**.** *Let $S = (X, U, \xrightarrow[S]{})$ be a metric transition system where $X \subseteq X_M$ for a metric space (X_M, d). Then the δ-**perturbed metric***

transition system *of S, $\mathfrak{S}^\delta$, is a tuple $\mathfrak{S}^\delta = (X, U, \xrightarrow[\mathfrak{S}^\delta]{})$ where the (altered) transition relation, $\xrightarrow[\mathfrak{S}^\delta]{}$, is defined as:*

$$x \xrightarrow[\mathfrak{S}^\delta]{u} x' \text{ iff } \exists x'' \in X \text{ s.t. } d(x'', x') \leq \delta \text{ and } x \xrightarrow[S]{u} x''. \quad (7)$$

Note that $\mathfrak{S}^\delta$ has identical states and input labels to S, and it also subsumes all of the transitions therein, i.e. $\xrightarrow[S]{} \subset \xrightarrow[\mathfrak{S}^\delta]{}$. However, the transition relation for $\mathfrak{S}^\delta$ explicitly contains new nondeterminism relative to the transition relation of S. This nondeterminism can be thought of as perturbing the target state of each transition in S; each such perturbation becomes the target of a (nondeterministic) transition with the same input label as the original transition.

Using this definition, we can finally define an abstract disturbance simulation between two metric transition systems.

Definition 8 (Abstract Disturbance Simulation)**.** *Let $S = (X_S, U, \xrightarrow[S]{})$ and $T = (X_T, U_T, \xrightarrow[T]{})$ be metric transition systems whose state spaces X_S and X_T are subsets of the same metric space (X_M, d). Then T* ***abstract-disturbance simulates*** *S under disturbance δ, written $S \preceq_{\mathcal{AD}_\delta} T$ if there is a relation $R \subseteq X_S \times X_T$ such that*

1) *for every $(x, y) \in R$, $d(x, y) \leq \delta$;*
2) *for every $x \in X_S$ there exists a pair $(x, y) \in R$; and*
3) *for every $(x, y) \in R$ and $x \xrightarrow[\mathfrak{S}^\delta]{u} x'$ there exists a $y \xrightarrow[T]{v} y'$ such that $(x', y') \in R$.*

Remark 1. *$\preceq_{\mathcal{AD}_0}$ corresponds with the usual notion of simulation for metric transition systems. Thus,*

$$S \preceq_{\mathcal{AD}_\delta} T \Leftrightarrow \mathfrak{S}^\delta \preceq_{\mathcal{AD}_0} T. \quad (8)$$

D. ReLU Neural Network Architectures

We will consider controlling the nonlinear system defined in (1) with a state-feedback neural network controller $\mathscr{NN}$:

$$\mathscr{NN} : X \to U \quad (9)$$

where $\mathscr{NN}$ denotes a Rectified Linear Unit Neural Network (ReLU NN). Such a (K-layer) ReLU NN is specified by composing K *layer* functions (or just *layers*). A layer with $\mathfrak{i}$ inputs and $\mathfrak{o}$ outputs is specified by a $(\mathfrak{o} \times \mathfrak{i})$ matrix of *weights*, W, and a $(\mathfrak{o} \times 1)$ matrix of *biases*, b, as follows:

$$L_\theta : \mathbb{R}^{\mathfrak{i}} \to \mathbb{R}^{\mathfrak{o}}, \quad z \mapsto \max\{Wz + b, 0\} \quad (10)$$

where the $\max$ function is taken element-wise, and $\theta \triangleq (W, b)$ for brevity. Thus, a K-layer ReLU NN function is specified by K layer functions $\{L_{\theta^{(i)}} : i = 1, \ldots, K\}$ whose input and output dimensions are *composable*: that is they satisfy $\mathfrak{i}_i = \mathfrak{o}_{i-1} : i = 2, \ldots, K$. Specifically:

$$\mathscr{NN}(x) = (L_{\theta^{(K)}} \circ L_{\theta^{(K-1)}} \circ \cdots \circ L_{\theta^{(1)}})(x). \quad (11)$$

When we wish to make the dependence on parameters explicit, we will index a ReLU function $\mathscr{NN}$ by a *list of matrices* $\Theta \triangleq (\theta^{(1)}, \ldots, \theta^{(K)})$ [1].

Specifying the number of layers and the *dimensions* of the associated matrices $\theta^{(i)} = (W^{(i)}, b^{(i)})$ specifies the *architecture* of the ReLU NN. Therefore, we will use:

$$\text{Arch}(\Theta) \triangleq ((n, \mathfrak{o}_1), (\mathfrak{i}_2, \mathfrak{o}_2), \ldots, (\mathfrak{i}_K, m)) \quad (12)$$

[1]That is Θ is not the concatenation of the $\theta^{(i)}$ into a single large matrix, so it preserves information about the sizes of the constituent $\theta^{(i)}$.

to denote the architecture of the ReLU NN $\mathscr{NN}_\Theta$.

Since we are interested in designing ReLU architectures, we will also need the following result from [7, Theorem 7], which states that a Continuous, Piecewise Affine (CPWA) function, f, can be implemented exactly using a Two-Level-Lattice (TLL) NN architecture that is parameterized exclusively by the number of local linear functions in f.

Definition 9 (Local Linear Function)**.** *Let $\mathsf{f} : \mathbb{R}^n \to \mathbb{R}^m$ be CPWA. Then a* ***local linear function of*** *f is a linear function $\ell : \mathbb{R}^n \to \mathbb{R}^m$ if there exists an open set $\mathfrak{O}$ such that $\ell(x) = \mathsf{f}(x)$ for all $x \in \mathfrak{O}$.*

Definition 10 (Linear Region)**.** *Let $\mathsf{f} : \mathbb{R}^n \to \mathbb{R}^m$ be CPWA. Then a* ***linear region of*** *f is the largest set $\mathfrak{R} \subseteq \mathbb{R}^n$ such that f has only one local linear function on int$(\mathfrak{R})$.*

Theorem 1 (Two-Level-Lattice (TLL) NN Architecture [7, Theorem 7])**.** *Let $\mathsf{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a CPWA function, and let $\bar{N}$ be an upper bound on the number of local linear functions in f. Then there is a Two-Level-Lattice (TLL) NN architecture Arch$(\Theta^{TLL}_{\bar{N}})$ parameterized by $\bar{N}$ and values of $\Theta^{TLL}_{\bar{N}}$ such that:*

$$\mathsf{f}(x) = \mathscr{NN}_{\Theta^{TLL}_{\bar{N}}}(x). \quad (13)$$

In particular, the number of linear regions of f is such an upper bound on the number of local linear functions.

Finally, note that a ReLU NN function, $\mathscr{NN}$, is known to be a continuous, piecewise affine (CPWA) function consisting of finitely many linear segments. Thus, $\mathscr{NN}$ is itself necessarily globally Lipschitz continuous.

III. Problem Formulation

We can now state the main problem that we will consider in this paper. In brief, we wish to identify the architecture for a ReLU network to be used as a feedback controller for the control system Σ: this architecture must have parameter weights that allow it to control Σ up to a specification that can be met by some other, non-NN controller.

Despite our choice to consider fundamentally continuous-time models, we formulate our main problem in terms of their (τ-sampled) transition system embeddings. This choice reflects recent success in verifying specifications for such transition system embeddings by means of techniques adapted from computer science; see e.g. [13], where a variety of specifications are considered in this context, among them LTL formula satisfaction. Thus, our main problem is stated in terms of the simulation relations in the previous section.

Problem 1. *Let $\delta > 0$ and $K_{cont} > 0$ be given. Let Σ be a feedback controllable Lipschitz control system, and let $S_{spec} = (X_{spec}, U_{spec}, \xrightarrow[S_{spec}]{})$ be a transition system encoding for a specification on Σ. Finally, let $\tau = \tau(K_x, K_u, \mathcal{K}, K_{cont}, \delta)$ be determined by the parameters specified.*

Now, suppose that there exists a δ,τ positively invariant Lipschitz-continuous controller $\Psi : \mathbb{R}^n \to U$ with Lipschitz constant $K_\Psi \leq K_{cont}$ such that:

$$S_\tau(\Sigma_\Psi) \preceq_{\mathcal{AD}_\delta} S_{spec}. \quad (14)$$

Then the problem is to find a ReLU architecture, Arch(Θ), with the property that there exists values for Θ such that:

$$S_\tau(\Sigma_{\mathscr{NN}_\Theta}) \preceq_{\mathcal{AD}_0} S_{spec}. \quad (15)$$

The main assumption in Problem 1 is that there exists a controller Ψ which satisfies the specification, S_{spec}. We use this assumption largely to help ensure that the problem is well posed. For example, this assumption ensures that we aren't trying to assert the existence of NN controller for a system and specification that can't be achieved by *any* continuous controller – such examples are known to exist for nonlinear systems. In this way, the existence of a controller Ψ subsumes any possible conditions of this kind that one might wish to impose: stabilizability, for example. Finally, note that the existence of such a Ψ may require knowledge of f to verify, but once its existence can be asserted the only explicit knowledge of f we assume is K_x, K_u and $\mathcal{K}$.

Moreover, there is a strong conceptual reason to consider abstract disturbance simulation in specification satisfaction for such a Ψ. Our approach to solve this problem will be to design a NN architecture that can approximate ***any*** such Ψ sufficiently closely. However, $\mathscr{NN}_\Theta$ clearly belongs to a smaller class of functions than Ψ, so an arbitrary controller Ψ cannot, in general, be represented *exactly* by means of $\mathscr{NN}_\Theta$. This presents an obvious difficulty because instantaneous errors between Ψ and $\mathscr{NN}_\Theta$ may accumulate by means of the system dynamics, i.e. via (4).

IV. ReLU Architectures for Nonlinear Systems

Before we state the main theorem of the paper, we introduce the following notation in the form of a definition.

Definition 11 (Extent of X)**.** *The **extent** of a compact set X is defined as:*

$$ext(X) \triangleq \max_{k=1,\dots,n} \left| \max_{x\in X} \pi_k(x) - \min_{x\in X} \pi_k(x) \right|, \tag{16}$$

where $\pi_k(x)$ is the projection of x onto its k^{th} component.

The main result of the paper is the following theorem, which directly solves Problem 1.

Theorem 2 (ReLU Architecture)**.** *Let $\delta > 0$ and $K_{cont} > 0$ be given, and let Σ and S_{spec} be as in the statement of Problem 1. Finally, choose a $\mu > 0$ such that:*

$$K_u \cdot \mu \cdot \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}} \cdot e^{K_x \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}}} < \delta, \tag{17}$$

and set:

$$\tau \leq \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}} \quad \text{and} \quad \eta \leq \frac{\mu}{6 \cdot K_{cont}}, \tag{18}$$

(which depend only on $\mathcal{K}$, K_x, K_u, K_{cont} and δ).

If there exists a δ, τ positively invariant Lipschitz continuous controller $\Psi : \mathbb{R}^n \to U$ with Lipschitz constant $K_\Psi \leq K_{cont}$ such that:

$$S_\tau(\Sigma_\Psi) \preceq_{\mathcal{AD}_\delta} S_{spec}. \tag{19}$$

Then a TLL NN architecture $Arch(\Theta_N^{TLL})$ of size:

$$N \geq m \cdot \left(n! \cdot \sum_{k=1}^{n} \frac{2^{2k-1}}{(n-k)!} \right) \cdot \left(\frac{ext(X)}{\eta} \right)^n \tag{20}$$

has the property that there exist values for Θ_N^{TLL} such that:

$$S_\tau(\Sigma_{\mathscr{NN}_{\Theta_N^{TLL}}}) \preceq_{\mathcal{AD}_0} S_{spec}. \tag{21}$$

Remark 2. *The left-hand side of (17) looks like μ^2 for small μ, so we can choose $\tau \approx \sqrt{\delta}$, thereby obtaining $\lim_{\delta\to 0} \frac{\tau}{\delta} = +\infty$. Thus, for δ small enough, the choice of τ in Theorem 2 is compatible with a δ, τ positively invariant controller (e.g. if $\|f\| > a > 0$ on $edge_{\delta'}(X)$ for some $\delta' > 0$).*

Proof Sketch:
The proof of Theorem 2 consists of establishing the following two implications:

Step 1) *"Approximate controllers satisfy the specification":* There is an approximation accuracy, μ, and sampling period, τ, with the following property: if the unknown controller Ψ satisfies the specification (under δ disturbance and sampling period τ), then any controller – NN or otherwise – which approximates Ψ to accuracy μ will also satisfy the specification (but under no disturbance). See Lemma 2 of Section V.

Step 2) *"Any controller can be approximated by a CPWA with the same fixed number of linear regions":* If unknown controller Ψ has a Lipschitz constant $K_\Psi \leq K_{\text{cont}}$, then Ψ can be approximated by a CPWA with a number of regions that depends only on K_{cont} and the desired approximation accuracy. See Lemma 4 of Section VI.

We will show these results for *any* controller Ψ that satisfies the assumptions of Theorem 2. Thus, these results together show the following implication: if there *exists* a controller Ψ that satisfies the assumptions of Theorem 2, then there is a CPWA controller that satisfies the specification. And moreover, this CPWA controller has at most a number of linear regions that depends only on the parameters of the problem *and not the particular controller* Ψ.

The conclusion of the theorem then follows directly from Theorem 1 [7, Theorem 7]: together, they specify that any CPWA with the same number of linear regions (or fewer) can be implemented exactly by a common TLL NN architecture.

V. Proof of Theorem 2, Step 1: Approximate Controllers Satisfy the Specification

The goal of this section is to choose constants $\mu > 0$ and $\tau > 0$ such that *any* controller Υ with $\|\Upsilon - \Psi\|_\infty \leq \mu/3$ satisfies the specification:

$$S_\tau(\Sigma_\Upsilon) \preceq_{\mathcal{AD}_0} S_{\text{spec}}. \tag{22}$$

The approach will be as follows. First, we confine ourselves to a region in the state space on which the controller Ψ doesn't vary much: the size of this region is determined entirely by the approximation accuracy, μ, and the bound on the Lipschitz constant, K_{cont}. Then we confine the *trajectories* of Σ_Ψ to this region by bounding the duration of those trajectories, i.e. τ. Finally, we feed these results into a Grönwall-type bound to choose μ. In particular, we choose μ small enough such that the error incurred by using Υ instead of Ψ is within the disturbance robustness, δ. From this we will conclude that Υ satisfies the specification whenever $\|\Upsilon - \Psi\| \leq \mu/3$. A road map of these steps is as follows.

- Let μ be an approximation error. Then:
  - *i)* Choose $\eta = \eta(\mu)$ such that a Lipschitz function with constant K_{cont} doesn't vary by more than $\mu/3$ between any two points that are 2η apart.
  - *ii)* Choose $\tau = \tau(\mu)$ such that $\|x - \xi_{xv}(\tau)\| \leq \eta$ for any continuous open-loop control v (use $\|f\| \leq \mathcal{K}$).

iii) Use i) and ii) to conclude that $\|\Upsilon(\zeta_{x\Upsilon}(t)) - \Psi(\zeta_{x\Psi}(t))\| \leq \|\Upsilon - \Psi\|_\infty + 2\mu/3$ for $t \in [0,\tau]$

iv) Assume $\|\Upsilon - \Psi\|_\infty \leq \mu/3$. Choose $\mu = \mu(\delta)$ such that a Grönwall-type bound satisfies:

$$\|\zeta_{x\Upsilon}(\tau(\mu)) - \zeta_{x\Psi}(\tau(\mu))\| \leq K_u \cdot \mu \cdot \tau(\mu) \cdot e^{K_x \tau(\mu)} < \delta. \quad (23)$$

Conclude that if $\|\Upsilon - \Psi\|_\infty \leq \mu/3$, then:

$$S_\tau(\Sigma_\Upsilon) \preceq_{\mathcal{AD}_0} \mathfrak{S}_\tau(\Sigma_\Psi) \preceq_{\mathcal{AD}_0} S_{\text{spec}}. \quad (24)$$

Full proofs of the following can be found in [14].

First, we formalize *i) - iii)* in the following propositions.

Proposition 1. *Let $\mu > 0$ be given, and let Ψ be as above. Then there exists an $\eta = \eta(\mu)$ such that:*

$$\|x - x'\| \leq 2\eta \implies \|\Psi(x) - \Psi(x')\| \leq \mu/3. \quad (25)$$

Proposition 2. *Let $\mu > 0$ be given, and let $\eta = \eta(\mu)$ be as in the previous proposition. Finally, let Σ be as specified in the statement of Theorem 2. Then there exists a $\tau = \tau(\mu)$ such that for any Lipschitz feedback controller Υ:*

$$\|x - \zeta_{x\Upsilon}(t)\| \leq \eta = \eta(\mu) \quad \forall t \in [0,\tau]. \quad (26)$$

Proposition 3. *Let $\mu > 0$ be given. Let Σ and Ψ be as in the statement of Theorem 2; let $\eta = \eta(\mu)$ be as in Proposition 1; let $\tau = \tau(\mu)$ be as in Proposition 2; and let $\Upsilon : \mathbb{R}^n \to U$ be a Lipschitz continuous function. Then:*

$$\forall t \in [0,\tau] \quad \|\Upsilon(\zeta_{x\Upsilon}(t)) - \Psi(\zeta_{x\Psi}(t))\| \leq \|\Upsilon - \Psi\|_\infty + \tfrac{2\mu}{3} \quad (27)$$

To prove Step *iv)* we first need the following two results.

Proposition 4 (Grönwall Bound)**.** *Let Σ and Ψ be as in the statement of Theorem 2, and let Υ be as in the statement of Proposition 3. If:*

$$\|\Upsilon(\zeta_{x\Upsilon}(t)) - \Psi(\zeta_{x\Psi}(t))\| \leq \kappa \quad \forall t \in [0,\tau] \quad (28)$$

then:

$$\|\zeta_{x\Upsilon}(t) - \zeta_{x\Psi}(t)\| \leq K_u \cdot \kappa \cdot t \cdot e^{K_x t} \quad \forall t \in [0,\tau]. \quad (29)$$

Lemma 1. *Let Σ, Ψ and Υ be as before. Also, suppose that $\mu > 0$ is such that:*

$$K_u \cdot \mu \cdot \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}} \cdot e^{K_x \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}}} < \delta. \quad (30)$$

If $\|\Upsilon - \Psi\|_\infty \leq \mu/3$, then:

$$\|\zeta_{x\Upsilon}(\tau(\mu)) - \zeta_{x\Psi}(\tau(\mu))\| \leq \delta. \quad (31)$$

The final result in this section is the following Lemma.

Lemma 2. *Let Σ, Ψ and Υ be as before, and suppose that $\mu > 0$ is such that:*

$$K_u \cdot \mu \cdot \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}} \cdot e^{K_x \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}}} < \delta. \quad (32)$$

If $\|\Upsilon - \Psi\|_\infty \leq \mu/3$, then for $\tau \leq \frac{\mu}{6 \cdot K_{cont} \cdot \mathcal{K}}$ we have:

$$S_\tau(\Sigma_\Upsilon) \preceq_{\mathcal{AD}_0} \mathfrak{S}_\tau(\Sigma_\Psi). \quad (33)$$

And hence:

$$S_\tau(\Sigma_\Upsilon) \preceq_{\mathcal{AD}_0} S_{spec}. \quad (34)$$

VI. Proof of Theorem 2, Step 2: CPWA Approximation of a Controller

The results in Section V showed that any controller, Υ, whether it is CPWA or not, will satisfy the specification if it is close to Ψ in the sense that $\|\Upsilon - \Psi\|_\infty \leq \mu/3$ (where μ is as specified therein). Thus, the main objective of this section will be to show that an arbitrary Ψ can be approximated to this accuracy by a CPWA controller, Υ_{CPWA}, subject to the following caveat. It is well known that CPWA functions are good function approximators in general, but we have to keep in mind our eventual use of Theorem 1: we need to approximate *any such* Ψ by a CPWA with the **same, bounded number of linear regions**. Hence, our objective in this section is to find not just a controller Υ_{CPWA} that approximates Ψ to the specified accuracy, but one that achieves this using not more than some common, fixed number of linear regions that depends only on the problem parameters (and not the function Ψ itself, which is assumed unknown except for a bound on its Lipschitz constant).

With this in mind, our strategy will be to partition the set X into a grid of sup-norm balls such that no relevant Ψ can vary by much between them: indeed, we will use balls of size η, as specified in Section V. Thus, we propose the following starting point: inscribe a slightly smaller ball within each η ball of the partition, and choose the value of Υ_{CPWA} on each such ball to be a constant value equal to $\Psi(x)$ for some x therein. Because we have chosen the size of the partition to be small, such an Υ_{CPWA} will still be a good approximation of Ψ for these points in its domain. Using this approach, then, we only have to concern ourselves with how "extend" a function so defined to the entire set X as a CPWA. Moreover, note that this procedure is actually independent of the particular Ψ chosen, despite appearances: we are basing our construction on a grid size η that depends only on the problem parameters (via μ), so the construction will work no matter the chosen value of $\Psi(x)$ within each grid square.

The first step in this procedure will be to show how to extend such a function over the largest-dimensional "gaps" between the smaller inscribed balls. This result must control the error of the extension so as to preserve our desired approximation bound, as well provide a count of the number of linear regions necessary to do so; this is Lemma 3. The preceding result can then be extended to all of the other gaps between inscribed balls to yield a CPWA function with domain X, approximation error $\mu/3$, and a known number of regions; this is Lemma 4. Full proofs appear in [14].

Definition 12 (Face/Corner)**.** *Let $C = [0,1]^n$ be a unit hypercube of dimension n. A set $F \subseteq C$ is a k-dimensional **face** of C if there exists a set $J \subseteq \{1,\ldots,n\}$ such that $|J| = n-k$ and*

$$\forall x \in F \;.\; \bigwedge_{j \in J} \pi_j(x) \in \{0,1\}. \quad (35)$$

*Let $\mathscr{F}_k(C)$ denote the set of k-dimensional faces of C, and let $\mathscr{F}(C)$ denote the set of all faces of C (of any dimension). A **corner** of C is a 0-dimensional face of C.*

Lemma 3. *Let $C = [0,1]^n$, and suppose that:*

$$\Gamma_c : \mathscr{F}_0(C) \to \mathbb{R} \quad (36)$$

is a function defined on the corners of C. Then there is a CPWA function $\Gamma : C \to \mathbb{R}$ such that:

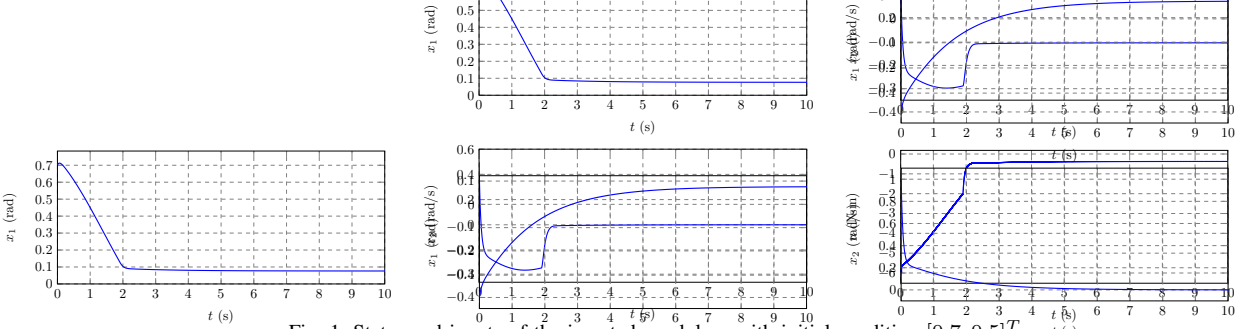

Fig. 1. States and inputs of the inverted pendulum with initial condition $[0.7, 0.5]^T$.

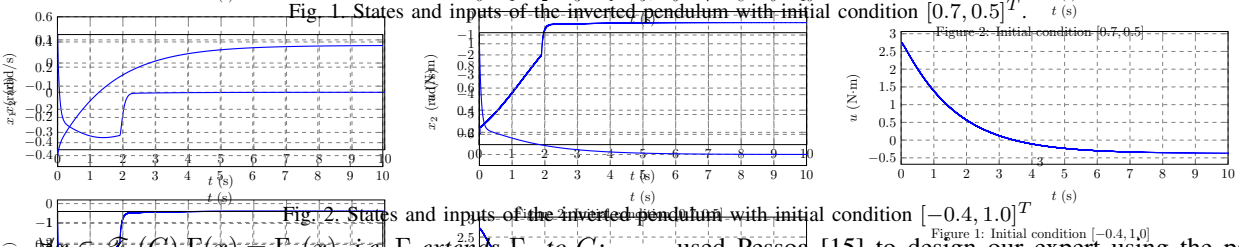

Fig. 2. States and inputs of the inverted pendulum with initial condition $[-0.4, 1.0]^T$

- $\forall x \in \mathscr{P}_0(C).\Gamma(x) = \Gamma_c(x)$, *i.e.* Γ *extends* Γ_c *to* C;
- Γ *has at most* $2^{n-1} \cdot n!$ *linear regions; and*
- *for all* $x \in C$,

$$\min_{x \in \mathscr{P}_0(C)} \Gamma_c(x) \leq \Gamma(x) \leq \max_{x \in \mathscr{P}_0(C)} \Gamma_c(x). \quad (37)$$

Lemma 4. *Let* $\eta = \eta(\mu)$ *be chosen as in Proposition 1, and let* Ψ *be as before. Then there is a CPWA function* $\Upsilon_{CPWA} : \mathbb{R}^n_u \to U$ *such that:*

- $\|\Upsilon_{CPWA} - \Psi\|_\infty \leq \frac{\mu}{3}$; *and*
- Υ_{CPWA} *has linear regions numbering at most*

$$m \cdot \left(n! \cdot \sum_{k=1}^{n} \frac{2^{2k-1}}{(n-k)!} \right) \cdot \left(\frac{ext(X)}{\eta} \right)^n. \quad (38)$$

VII. Numerical Results

We illustrate the results in this paper on an inverted pendulum described by the following model:

$$f(x_1, x_2, u) = \begin{bmatrix} x_2 & \frac{g}{l}\sin(x_1) - \frac{h}{ml^2}x_2 + \frac{1}{ml}\cos(x_1)u \end{bmatrix}^{\mathrm{T}},$$

where x_1 is the angular position, x_2 is the angular velocity, and control input u is the torque applied on the point mass. The parameters are the rod mass, m; the rod length, l; the (dimensionless) coefficient of rotational friction, h; and the acceleration due to gravity, g. For the purposes of our experiments, we considered a subset of the state/control space specified by: $x_1 \in [-1, 1]$, $x_2 \in [-1, 1]$ and $u \in [-6, 6]$. Furthermore, we considered model parameters: $m = 0.5$ kg; $l = 0.5$ m; $h = 2$; and $g = 9.8$ N/kg. Then for different choices of the design parameters μ, we obtained the following table of sizes, N, for the corresponding TLL-NN architecture; also shown are the corresponding τ, η and the δ that are required for the specification satisfaction.

μ	δ	τ	η	N
0.35	0.8694	0.0098	0.583	235
0.3	0.5287	0.0083	0.5	320
0.25	0.3039	0.0069	0.417	460
0.2	0.1610	0.0056	0.334	720
0.15	0.0749	0.0042	0.25	1280
0.1	0.0275	0.0028	0.167	2880

In the sequel, we will show the control performance of a TLL-NN architecture with 400 local linear region. While there are a number of techniques that can be used to train the resulting NN, for simplicity, we utilize Imitation learning where the NN is trained in a supervised fashion from data collected from an expert controller. In particular, we designed an expert controller that stabilizes the inverted pendulum; we used Pessoa [15] to design our expert using the parameter values specified above. In particular, we tasked Pessoa to design a zero-order-hold controller that stabilizes the inverted pendulum in a subset $X_{\text{spec}} = [-1, 1] \times [-0.5, 0.5]$: i.e. the controller should transfer the state of the system to this specified set and keep it there for all time thereafter. From this expert controller, we collected 8400 data points of state-action pairs; this data was obtained by uniformly sampling the state space. We then used Keras [16] to train the TLL NN using this data. Finally, we simulated the motion of the inverted pendulum using this TLL NN controller. Shown in Fig. 1 and Fig. 2 are the state and control trajectories for this controller starting from initial state $[0.7, 0.5]$ and $[-0.4, 1]$, respectively. In both, the TLL controller met the same specification used to design the expert.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[2] F. Pedregosa, "Hyperparameter optimization with approximate gradient," *arXiv:1602.02355*, 2016.

[3] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[4] S. Paul, V. Kurin, and S. Whiteson, "Fast efficient hyperparameter tuning for policy gradients," *arXiv:1902.06583*, 2019.

[5] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv:1611.02167*, 2016.

[6] Y. Quanming, W. Mengshuo, J. E. Hugo, G. Isabelle, H. Yi-Qi, L. Yu-Feng, T. Wei-Wei, Y. Qiang, and Y. Yang, "Taking human out of learning applications: A survey on automated machine learning," *arXiv:1810.13306*, 2018.

[7] J. Ferlez and Y. Shoukry, "AReN: Assured ReLU NN Architecture for Model Predictive Control of LTI Systems," in *Hybrid Systems: Computation and Control 2020 (HSCC'20)*, ACM, New York, 2020.

[8] H. K. Khalil, *Nonlinear Systems*. Pearson, Third ed., 2001.

[9] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic Models for Nonlinear Control Systems Without Stability Assumptions," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, 2012.

[10] V. Kurtz, P. M. Wensing, and H. Lin, "Robust Approximate Simulation for Hierarchical Control of Linear Systems under Disturbances," 2020.

[11] K. Mallik, A.-K. Schmuck, S. Soudjani, and R. Majumdar, "Compositional Synthesis of Finite-State Abstractions," *IEEE Transactions on Automatic Control*, vol. 64, no. 6, pp. 2629–2636, 2019.

[12] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[13] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer US, 2009.

[14] J. Ferlez, X. Sun, and Y. Shoukry, "Two-Level Lattice Neural Network Architectures for Control of Nonlinear Systems," *arXiv:2004.09628*, 2020.

[15] M. Mazo, A. Davitian, and P. Tabuada, "PESSOA: A tool for embedded controller synthesis," in *Proceedings of the 22nd International Conference on Computer Aided Verification*, CAV'10, pp. 566–569, Springer-Verlag, 2010.

[16] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.