On the Computation of ReLU-based RNNs equivalent to CPWA models of Dynamical Systems and vice versa ⋆

Marco Ledda * **Diego Deplano** * **Alessandro Giua** *
Mauro Franceschelli *

* *DIEE, University of Cagliari, 09123 Cagliari, Italy.*

Abstract: This paper develops computational procedures to translate between ReLU recurrent neural networks (RNNs) and continuous piecewise-affine (CPWA) dynamical systems. We show that every ReLU RNN induces a finite polyhedral partition of the state–input space with affine dynamics on each region and provide an explicit constructive algorithm to compute its corresponding CPWA representation. Conversely, we show that every CPWA dynamical system admits an exact realization as a ReLU RNN and provide a constructive procedure to compute the associated network parameters. The computational procedures are illustrated by numerical examples for both directions, showcasing exactly matching trajectories and validating the implementation of the proposed transformations.

Keywords: Piecewise affine dynamical systems, Recurrent Neural Networks

1. INTRODUCTION

Hybrid dynamical systems provide a rigorous and unifying framework for describing processes where continuous-time dynamics interact with discrete events, logic-based switching, and physical constraints (Labinaz et al., 1997; Bemporad et al., 2001). These systems naturally arise in many engineering domains, including robotics, automotive systems, process control, and power electronics, where hybrid phenomena emerge from logical supervision, actuator saturation, and on–off control policies. Over the past two decades, several mathematical formalisms have been proposed to capture hybrid behaviors, including hybrid automata (Alur et al., 1995; Henzinger et al., 1998), mixed logical dynamical (MLD) systems (Bemporad and Morari, 1999; Karaman et al., 2008), and piecewise-affine (PWA) models (Heemels et al., 2001). Fundamental equivalence results among these representations have enabled the systematic transfer of analysis and control methods, providing a solid foundation for algorithmic controller synthesis, reachability analysis, and verification (Heemels et al., 2001).

Among the several hybrid representations, *piecewise-affine* (PWA) models play a central role due to their analytic tractability and compatibility with convex optimization (Biswas et al., 2005). In a PWA system, the state evolves under affine dynamics that depend on the current regions of a polyhedral partition of the state–input space. This structure allows direct application of linear programming tools for controller synthesis (Collins et al., 2009; Seatzu et al., 2006b), stability verification (Ferrari-Trecate et al., 2002; Lazar and Heemels, 2008), and robustness analysis (Rodrigues and Boyd, 2005; Ledda et al., 2025). Moreover, PWA models from the foundation of explicit model predictive control (MPC), where the optimal control law is computed offline as a set of affine feedbacks defined over polyhedral regions, ensuring real-time feasibility on embedded hardware (De Schutter and van den Boom, 2004; Seatzu et al., 2006a).

Beyond their control-oriented utility, PWA maps are also universal approximators of nonlinear functions: any continuous nonlinearity can be represented to arbitrary accuracy by a sufficiently refined PWA decomposition. In parallel, the widespread adoption of *neural networks* for system modeling and control (Masti and Bemporad, 2021) has brought renewed attention to architectures based on the Rectified Linear Unit (ReLU). Each ReLU introduces a linear region separated by an activation hyperplane, implying that any ReLU network realizes a continuous PWA function (Montúfa et al., 2014). Hence, feedforward or recurrent ReLU networks can be regarded as data–driven generators of high–dimensional PWA dynamics (Arora et al., 2018).

Recent works have explored the extraction of explicit PWA approximations from trained neural networks (Roll et al., 2004), aiming to combine learning flexibility with hybrid system interpretability. Conversely, the inverse problem—realizing an exact neural representation of a given CPWA mapping—has been addressed only under restrictive conditions, typically for one-dimensional inputs and limited segment counts (Darup, 2020; Dereich and Kassing, 2022).

⋆ The work of Diego Deplano was supported by the project e.INS- Ecosystem of Innovation for Next Generation Sardinia (cod. ECS 00000038) funded by the Italian Ministry for Research and Education (MUR) under the National Recovery and Resilience Plan (NRRP) - MISSION 4 COMPONENT 2, "From research to business" INVESTMENT 1.5, "Creation and strengthening of Ecosystems of innovation" and construction of "Territorial R& D Leaders". This work is partially supported by Fondazione di Sardegna with project "ReSNetS" call 2025, F83C26000350007.
E-mails: marco.ledda2@unica.it, diego.deplano@unica.it, giua@unica.it, mauro.franceschelli@unica.it.

© 2026 the authors. This work has been accepted to IFAC for publication under a Creative Commons Licence CC-BY-NC-ND.

This paper provides rigorous and constructive algorithms to translate between ReLU recurrent neural networks and discrete-time continuous piecewise-affine (CPWA) dynamical systems. The **main contributions** are:

- We establish the equivalence between the class of CPWA dynamical systems and that of ReLU RNNs (Thm. 3).
- We design a constructive algorithm to compute the CPWA dynamical system associated with any ReLU RNN, supported by the proof of Thm. 6 and detailed in Algorithm 1;
- We design a constructive algorithm to compute a ReLU RNN realization of any CPWA dynamical system, supported by the proof of Thm. 7 and detailed in Algorithm 2.

Structure of the paper. Section 2 introduces notation and definitions of CPWA systems and ReLU RNNs. Section 3 presents the equivalence results and the algorithms for both transformations. Section 4 reports numerical validations, while Section 5 concludes the paper.

2. NOTATION AND PRELIMINARIES ON PWA DYNAMICAL SYSTEMS AND RNNS

A piecewise affine (PWA) dynamical system is characterized by a state space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, an input space $\mathcal{U} \subseteq \mathbb{R}^{n_u}$, and a partition $\Omega := \mathcal{X} \times \mathcal{U} = \Omega_1 \cup \cdots \cup \Omega_m$ of the space into $m \in \mathbb{N}$ convex polyhedral regions (possibly open) each with an affine dynamics, namely

$$x_{k+1} = f_{\text{PWA}}(x_k, u_k) := \begin{cases} A_1 x_k + B_1 u_k + c_1 & \text{if } (x_k, u_k) \in \Omega_1, \\ \vdots & \\ A_m x_k + B_m u_k + c_m & \text{if } (x_k, u_k) \in \Omega_m. \end{cases} \tag{1}$$

Definition 1. Consider a PWA dynamical system with state-update rule f_{PWA} as in eq. (1). If the values of f_{PWA} coincide at all points in the intersection of the closures of any two adjacent regions Ω_i, Ω_j, then (1) defines a continuous PWA (CPWA) dynamical system, namely, $\forall i, j \in \{1, \cdots, m\}$:

$$A_i x + B_i u + c_i = A_j x + B_j u + c_j, \quad \forall (x, u) \in \text{cl}(\Omega_i) \cap \text{cl}(\Omega_j).$$

A RNN (depicted in Fig. 1) is characterized by a state space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, an input space $\mathcal{U} \subseteq \mathbb{R}^{n_u}$, and a sequence of $L \in \mathbb{N}$ layers, each with hidden update rule $f_\ell : \mathbb{R}^{n_{\ell-1}} \to \mathbb{R}^{n_\ell}$ with $n_\ell \in \mathbb{N}$ denoting the number of neurons for $\ell \in \{1, \cdots, L\}$, namely

$$x_{k+1} = f_{\text{RNN}}(x_k, u_k) := f_L(f_{L-1}(\cdots(f_1(x_k, u_k)))). \tag{2}$$

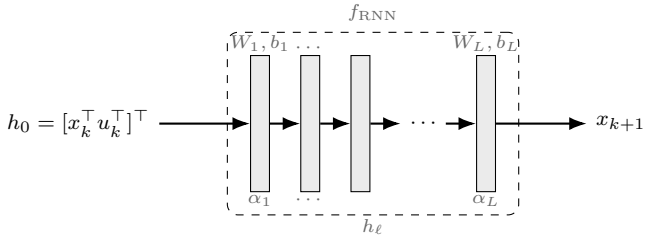


Fig. 1. Schematic of the recurrent neural network in Def. 2. The state x_k is propagated through L parameterized layers along with the input u_k to return the next state x_{k+1}.

Let the output vector of the ℓ-th layer be denoted by $h_\ell := f_\ell(h_{\ell-1}) \in \mathbb{R}^{n_\ell}$ with $h_0 = [x_k^\top, u_k^\top]^\top$, $n_0 = n_x + n_u$, $n_L = n_x$ and $h_L = x_{k+1}$. In particular, we consider RNNs such that each f_ℓ computes an affine combination of the outputs of the previous layer, i.e., $W_\ell h_{\ell-1} + b_\ell$ where $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $b_\ell \in \mathbb{R}^{n_\ell}$ are the weight matrix and bias vector. Optionally, the result of the affine operation can be transformed by the *Rectified Linear Unit (ReLU)* activation function, defined element-wise as

$$(\text{ReLU}(v))_i := \max(0, v_i), \qquad i = 1, \ldots, n, \tag{3}$$

for any vector $v \in \mathbb{R}^n$. Hence, ReLU$(\cdot)$ acts as a mapping that preserves the non-negative components of its argument and sets the others to zero. A more general form used in this paper is the *Leaky–ReLU* function,

$$(\text{LReLU}_\alpha(v))_i := \alpha v_i + (1 - \alpha)\,\text{ReLU}(v_i), \tag{4}$$

where $\alpha \in [0, 1]$. When $\alpha = 0$ it coincides with the standard ReLU, and when $\alpha = 1$, it reduces to the identity map.

Definition 2. Consider a RNN with state-update rule f_{RNN} as in eq. (2). If f_{RNN} is an arbitrary composition of affine and LReLU transformations of the form in eq. (4), then (2) defines a ReLU RNN, namely,

$$f_\ell(h_{\ell-1}) = \text{LReLU}_{\alpha_\ell}(W_\ell h_{\ell-1} + b_\ell), \tag{5}$$

for $\ell = 1, \ldots, L$, with $\alpha_\ell \in [0, 1]$ indicating whether layer ℓ acts as a ReLU ($\alpha_\ell = 0$) or an affine mapping ($\alpha_\ell = 1$).

Let $\mathcal{N} = \big\{(\ell, \eta) \mid \ell \in \{1, \cdots, L\}, \eta \in \{1, \cdots, n_\ell\}\big\}$ be the set representing all neurons in the RNN, and denote the subset of neurons with a ReLU activation function by

$$\widetilde{\mathcal{N}} := \big\{(\ell, \eta) \in \mathcal{N} \mid \alpha_\ell = 0\big\} \subseteq \mathcal{N} \tag{6}$$

The function $\varsigma : \widetilde{\mathcal{N}} \times \Omega \to \{0, 1\}$ describing the activation pattern of the RNN neurons is defined as follows

$$\varsigma(\ell, \eta, x, u) = \begin{cases} 1, & \text{if } (f_\ell(\cdots(f_1(x, u)))_\eta > 0, \\ 0, & \text{if } (f_\ell(\cdots(f_1(x, u)))_\eta \leq 0. \end{cases}$$

The activation pattern for the ℓ-th nonlinear layer such that $\alpha_\ell < 1$ is denoted by $\sigma_\ell(x, u) \in \{0, 1\}^{n_\ell}$, obtained by concatenating all $\varsigma(\ell, \eta, x, u)$ for $\eta = 1, \ldots, n_\ell$. Consequently, the activation pattern for the whole RNN is $\sigma(x, u) \in \{0, 1\}^{|\widetilde{\mathcal{N}}|}$, obtained by concatenating all $\sigma_\ell(x, u)$ for $\ell = 1, \ldots, L$ such that $\alpha_\ell < 1$. For any fixed $\sigma^\star \in \{0, 1\}^{|\widetilde{\mathcal{N}}|}$, the *activation region* $\mathcal{R}(\sigma^\star) \subseteq \Omega$ is given by

$$\mathcal{R}(\sigma^\star) := \{(x, u) \in \Omega \mid \sigma(x, u) = \sigma^\star\}.$$

3. EQUIVALENCE TO PIECEWISE AFFINE DYNAMICAL SYSTEM

We next recall the known equivalence between ReLU-based RNNs and CPWA dynamical systems, providing an original proof based on the practical construction, designed in this paper, of each representation from the other.

Theorem 3. The two following classes are equivalent:

(i) CPWA dynamical systems as in Def. 1.
(ii) ReLU RNN as in Def. 2.

Proof. The implication (i) $\Leftarrow$ (ii) follows by Thm. 6 in Section 3.1, while the implication (i) $\Rightarrow$ (ii) follows by Thm. 7 in Section 3.2. □

© 2026 the authors. Licensed under CC-BY-NC-ND.

3.1 From ReLU RNNs to CPWA dynamical systems

We now show that every RNN with ReLU activations can be equivalently represented by a CPWA dynamical system (see Thm. 6). Algorithm 1 details the procedure to design a CPWA dynamical system that is equivalent to a given ReLU RNN by following the constructive proof of Thm. 6, which requires two intermediate results: each activation region of the RNN is a polyhedral region (see Lemma 4); each region the state-update map is affine (see Lemma 5).

Lemma 4. Consider an RNN as in Def. 2. For each activation pattern $\sigma^\star = [\sigma_1^\star; \cdots; \sigma_L^\star] \in \{0,1\}^{|\widetilde{\mathcal{N}}|}$, the activation region $\mathcal{R}(\sigma^\star) \subseteq \Omega$ is a polyhedral region determined by the following set of inequalities:

$$\mathcal{R}(\sigma^\star)=\left\{(x,u)\in\Omega \,\middle|\, \forall(\ell,\eta)\in\widetilde{\mathcal{N}}: \begin{cases} H_{\ell,\eta}^\top \begin{bmatrix} x \\ u \end{bmatrix} > q_{\ell,\eta}, \text{ if } \sigma^\star_{\ell,\eta}=1, \\ H_{\ell,\eta}^\top \begin{bmatrix} x \\ u \end{bmatrix} \leq q_{\ell,\eta}, \text{ otherwise} \end{cases} \right\} \quad (7)$$

where the inequalities are defined by using the η-th row of W_ℓ denoted by $W_{\ell,\eta}^\top \in \mathbb{R}^{1\times n_{\ell-1}}$ and the matrix $D_\ell = \operatorname{diag}(\alpha_\ell \mathbb{1}_{n_\ell} - (\alpha_\ell - 1)\sigma_\ell^\star)$, as follows

$$H_{\ell,\eta}^\top = W_{\ell,\eta}^\top \left(\prod_{i=1}^{\ell-1} D_i W_i\right), \quad (8)$$

$$q_{\ell,\eta} = -b_{\ell,\eta} - \sum_{i=1}^{\ell-1} \Big(W_{\ell,\eta}^\top \prod_{j=i+1}^{\ell-1} D_j W_j\Big) D_i b_i. \quad (9)$$

Proof. We prove the statement by induction on the layer index ℓ. Let $h_0 \in \Omega$ denote the input of the RNN. For each layer $\ell = 1, \ldots, L$, it holds

$$h_\ell = \mathrm{LReLU}_{\alpha_\ell}(W_\ell h_{\ell-1} + b_\ell) = D_\ell(W_\ell h_{\ell-1} + b_\ell),$$

where D_ℓ is defined as in Lemma 4 and represent the fixed activation pattern σ_ℓ. For the first layer, it holds

$$h_1 = D_1 W_1 h_0 + D_1 b_1.$$

Substituting into the second layer yields

$$\begin{aligned} h_2 &= D_2(W_2 h_1 + b_2) \\ &= D_2 W_2 D_1 W_1 h_0 + D_2 W_2 D_1 b_1 + D_2 b_2. \end{aligned}$$

By recursive substitution, each layer output can be expressed as an affine function of the input vector h_0:

$$h_\ell = \left(\prod_{i=1}^{\ell} D_i W_i\right) h_0 + \sum_{i=1}^{\ell} \left(\prod_{j=i+1}^{\ell} D_j W_j\right) D_i b_i, \quad (10)$$

where the empty product is the identity matrix, by convention. Each layer ℓ such that $\alpha_\ell = 1$, i.e., it is a purely linear layer, no constraints need to be considered as no neuron is switched off by the activation function. Instead, for each layer ℓ such that $\alpha_\ell \in [0,1)$, the η-neuron is either switched on if $W_{\ell,\eta}^\top h_{\ell-1} + b_{\ell,\eta} > 0$ or off if $W_{\ell,\eta}^\top h_{\ell-1} + b_{\ell,\eta} \leq 0$.

By induction on h_ℓ, we complete the proof by rewriting $W_{\ell,\eta}^\top h_{\ell-1} + b_{\ell,\eta}$ in terms of $h_0 = [x^\top, u^\top]^\top$ as follows:

$$\underbrace{W_{\ell,\eta}^\top \left(\prod_{i=1}^{\ell-1} D_i W_i\right)}_{H_{\ell,\eta}^\top} h_0 + \underbrace{b_{\ell,\eta} + \sum_{i=1}^{\ell-1} \left(W_{\ell,\eta}^\top \prod_{j=i+1}^{\ell-1} D_j W_j\right) D_i b_i}_{q_{\ell,\eta}}.$$

This concludes the proof. □

Algorithm 1 From ReLU RNN to CPWA system

Input: Matrix weights W_ℓ, biases b_ℓ, activation indicators α_ℓ for $\ell = 1, \cdots, L$.
Output: Regions Ω_i and dynamics A_i, B_i, c_i, $\forall i \in [1, m]$.

```
Compute 𝒩̃ as in eq. (6)
for each (ℓ, η) ∈ 𝒩̃
    Compute H_{ℓ,η}^⊤ as in eq. (8)
    Compute q_{ℓ,η} as in eq. (9)
end for
Set the current region index at i = 1
for each activation pattern σ* ∈ {0,1}^{|𝒩̃|}
    Compute the region Ω = R(σ*) as in eq. (7)
    if Ω ≠ ∅
        Assign Ω_i = Ω
        Compute A_{σ*}, B_{σ*}, c_{σ*} as in eq. (11)
        Assign A_i = A_{σ*}, B_i = B_{σ*}, c_i = c_{σ*}
        Increment the region index i = i + 1
    end if
end for
Compute the total number of regions m = i
```

Lemma 5. Consider an RNN as in Def. 2. For each activation pattern $\sigma^\star = [\cdots; \sigma^\star_{\ell,\eta}; \cdots] \in \{0,1\}^{|\widetilde{\mathcal{N}}|}$, the state-update on the corresponding $\mathcal{R}(\sigma^\star)$ is an affine map uniquely determined by $x_{k+1} = A_{\sigma^\star} x_k + B_{\sigma^\star} u_k + c_{\sigma^\star}$ where

$$[A_{\sigma^\star} B_{\sigma^\star}] := \prod_{\ell=1}^{L} D_\ell W_\ell, \; c_{\sigma^\star} := \sum_{\ell=1}^{L} \left(\prod_{j=\ell+1}^{L} D_j W_j\right) D_\ell b_\ell, \quad (11)$$

Proof. From Lemma 4, the activations within a fixed pattern $\sigma^\star$ satisfy

$$h_\ell = D_\ell(W_\ell h_{\ell-1} + b_\ell)$$

with $h_0 = [x^\top \; u^\top]^\top$ and diagonal masks D_ℓ determined by $\sigma^\star$. Following the proof strategy of Lemma 4, the output of the ℓ-th layer is given by

$$h_\ell = \left(\prod_{i=1}^{\ell} D_i W_i\right) h_0 + \sum_{r=1}^{\ell} \left(\prod_{j=r+1}^{\ell} D_j W_j\right) D_r b_r.$$

The proof is completed by noticing that on $\mathcal{R}(\sigma^\star)$, the network output is affine in $h_0 = [x_k^\top, u_k^\top]^\top$ on each activation region $\mathcal{R}(\sigma^\star)$:

$$x_{k+1} = \underbrace{\left(\prod_{\ell=1}^{L} D_\ell W_\ell\right)}_{[A_\sigma \; B_\sigma]} h_0 + \underbrace{\sum_{\ell=1}^{L} \left(\prod_{j=\ell+1}^{L} D_j W_j\right) D_\ell b_\ell}_{c_{\sigma^\star}}.$$

This concludes the proof. □

Theorem 6. Every ReLU RNN as in Def. 2 can be equivalently written as a CPWA dynamical system as in Def. 1.

Proof. By Lemma 4, each activation pattern $\sigma^\star \in \{0,1\}^{|\mathcal{N}|}$ of the ReLU RNN determines a convex polyhedral region $\mathcal{R}(\sigma^\star) \subseteq \Omega$, constructed from the network weights and biases. The feasible nonempty regions form a finite polyhedral partition of Ω. Lemma 5 establishes that within any such region $\mathcal{R}(\sigma^\star)$ the network mapping is affine in x, u, with matrices $A_{\sigma^\star}$, $B_{\sigma^\star}$ and offset $c_{\sigma^\star}$ obtained from the recursive propagation of weights and biases. Continuity of the ReLU function ensures continuous dynamics across adjacent regions. □

© 2026 the authors. Licensed under CC-BY-NC-ND.

3.2 From CPWA dynamical systems to ReLU RNNs

We now show that every CPWA dynamical system can be equivalently represented by an RNN with ReLU activations (see Thm. 7). Algorithm 2 details the procedure to design a ReLU RNN that is equivalent to a given CPWA dynamical system by following the constructive proof of Thm. 7

Theorem 7. Every CPWA dynamical system as in Def. 1 can be equivalently written as a ReLU RNN as in Def. 2.

Proof. We start by denoting by $f^i_{\text{PWA}} : \mathbb{R}^{n_0} \to \mathbb{R}$ the i-th component of $f_{\text{PWA}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_x}$, namely:

$$f_{\text{PWA}}(x,u) = \begin{bmatrix} \vdots \\ f^i_{\text{PWA}}(x,u) \\ \vdots \end{bmatrix}, \quad n_0 := n_x + n_u.$$

Any CPWA real-valued function on a convex set can be equivalently represented as the difference of two convex CPWA functions (Kazda and Li, 2021). We decompose each f^i_{PWA} by:

$$f^i_{\text{PWA}}(x,u) := f^{i,+}_{\text{PWA}}(x,u) - f^{i,-}_{\text{PWA}}(x,u),$$

where $f^{i,\pm}_{\text{PWA}}(x,u)$ are convex CPWA functions (Hempel et al., 2015, Lemma 1), i.e.,

$$f^{i,\pm}_{\text{PWA}}(x,u) := \begin{cases} H^\pm_{i,1}[x^\top \;\; u^\top]^\top + q^\pm_{i,1} & \text{if } (x,u) \in \Omega_1, \\ \vdots \\ H^\pm_{i,m}[x^\top \;\; u^\top]^\top + q^\pm_{i,m} & \text{if } (x,u) \in \Omega_m. \end{cases}$$

By convexity, they can be equivalently written as the maximum of the corresponding m affine functions (cfr. with (Magnani and Boyd, 2009, Section 1.1)), as follows:

$$f^{i,\pm}_{\text{PWA}}(x,u) = \max_{j=1,\dots,m} \left\{ H^\pm_{i,j} \begin{bmatrix} x \\ u \end{bmatrix} + q^\pm_{i,j} \right\}. \tag{12}$$

where $H^\pm_{i,j} \in \mathbb{R}^{1\times n_0}$, $q^\pm_{i,j} \in \mathbb{R}$. Let

$$e^\pm_{i,j}(x,u) = H^\pm_{i,j}[x^\top \;\; u^\top]^\top + q^\pm_{i,j} \tag{13}$$

be one of the elements in the maximum operation; in the following, we omit the argument (x,u) for the sake of presentation. The maximum among the m elements can be decomposed into $m-1$ nested maxima between two elements, namely $f^{i,\pm}_{\text{PWA}} := e^{\pm,\star}_{i,m}$ with $e^{\pm,\star}_{i,1} = e^\pm_{i,1}$ and

$$e^{\pm,\star}_{i,\ell} = \max\{e^\pm_{i,\ell},\, e^{\pm,\star}_{i,\ell-1}\}, \quad \forall \ell = 2,\cdots,m. \tag{14}$$

Moreover, each maximum between two elements can be written in terms of ReLU and affine functions as follows:

$$\begin{aligned} &\max\{e^\pm_{i,j}, e^{\pm,\star}_{i,j-1}\} = \\ &\text{ReLU}(e^\pm_{i,j} - e^{\pm,\star}_{i,j-1}) + \text{ReLU}(e^{\pm,\star}_{i,j-1}) - \text{ReLU}(-e^{\pm,\star}_{i,j-1}). \end{aligned} \tag{15}$$

Thus, each binary maximum can be expressed by two layers, where the first one computes the ReLU$(\cdot)$ terms and the second performs the algebraic summation. By nesting these blocks, each $f^{i,\pm}_{\text{PWA}}(x,u)$ can be represented by a ReLU neural network. Repeating this process for all $i = 1,\cdots,n_x$ completes the proof. □

Remark 8. The computation of the vectors $H^\pm_{i,j}$ and $q^\pm_{i,j}$ can be carried out for general CPWA functions by following the constructive method outlined in the proof of (Hempel et al., 2015, Lemma 4). In the special case where the CPWA function is defined over a regular subdivision, such as the one considered in our numerical example in Section 4.2, the vectors $H^\pm_{i,j}$ and $q^\pm_{i,j}$ can be computed using the optimization problem described next, as discussed in (Hempel et al., 2015, III.B) Let us define the set of pairs (j,ℓ) indicating neighboring regions $\mathcal{I} = \{(j,\ell) \in \mathbb{N}\times\mathbb{N} \mid \dim(\,\text{cl}(\Omega_j) \cap \text{cl}(\Omega_\ell)) = n_x + n_u - 1\}$. Assuming a halfspace representation $\Omega_j = \{[x^\top \;\; u^\top]^\top \in \mathbb{R}^{n_x+n_u} \mid C_j[x^\top \;\; u^\top]^\top \le d_j\}$, coefficients $(H^\pm_{i,j}, q^\pm_{i,j})$ can be obtained by solving the following optimization problem

$$\begin{aligned} \min \quad & \sum_{j=1}^m \left(\|H^\pm_{i,j}\|_1 + |q^\pm_{i,j}| \right) \\ \text{s.t.} \quad & [A_j \;\; B_j] = H^+_{i,j} - H^-_{i,j}, && \forall j \in \{1,\cdots,m\}, \\ & c_j = q^+_{i,j} - q^-_{i,j}, && \forall j \in \{1,\cdots,m\}, \\ & C_j^\top \lambda^+_{j,\ell} = (H^+_{i,\ell} - H^+_{i,j})^\top && \forall (j,\ell) \in \mathcal{I}, \\ & C_j^\top \lambda^-_{j,\ell} = (H^-_{i,\ell} - H^-_{i,j})^\top && \forall (j,\ell) \in \mathcal{I}, \\ & d_j^\top \lambda^+_{j,\ell} \le q^+_{i,j} - q^+_{i,\ell} && \forall (j,\ell) \in \mathcal{I}, \\ & d_j^\top \lambda^-_{j,\ell} \le q^-_{i,j} - q^-_{i,\ell} && \forall (j,\ell) \in \mathcal{I}, \end{aligned} \tag{16}$$

where $\lambda^\pm_{j,\ell}$ are Lagrangian multipliers. □

Algorithm 2 From CPWA system to ReLU RNN

Input: Dynamics A_r, B_r, c_r and regions Ω_r $\forall r \in [1,m]$.
Output: Matrix weights W_ℓ, biases b_ℓ, activation indicators α_ℓ for $\ell = 1,\cdots,L$

```
for each state component i = 1, ..., n_x
    Write e^{±,⋆}_{i,1}(x,u) := e^±_{i,1}(x,u)
    for j = 2, ..., m
        Compute H^±_{i,j}, q^±_{i,j} as discussed in Rem. 8
        Write e^±_{i,j}(x,u) as in eq. (13)
        Write e^{±,⋆}_{i,j}(x,u) as in eq. (14)
        Write max{e^±_{i,j}(x,u), e^{±,⋆}_{i,j-1}(x,u)} as in eq. (15)
    end for
    Write f^{i,±}_PWA(x,u) := e^{±,⋆}_{i,m}(x,u)
    Write f^i_PWA(x,u) := f^{i,+}_PWA(x,u) − f^{i,−}_PWA(x,u)
    Determine {W_{ℓ,i}, b_{ℓ,i}}_{ℓ=1}^{2(m−1)} based on the expression obtained for f^i_PWA(x,u)
end for
Compute W_ℓ = [W^⊤_{ℓ,1}, ..., W^⊤_{ℓ,n_x}]^⊤
Compute b_ℓ = [b^⊤_{ℓ,1}, ..., b^⊤_{ℓ,n_x}]^⊤
Set α_ℓ = 1 for odd values of ℓ and 0 otherwise
```

4. NUMERICAL VALIDATION

The code to reproduce the numerical simulations is available at `https://github.com/mledda17/rnn_pwa`.

4.1 Numerical example from ReLU RNN to CPWA

We consider a RNN composed of two layers. The first layer consists of 3 ReLU neurons, and the second layer comprises two linear neurons (without an activation function). This last layer is linear. The state and input dimensions are $n_x = 2$, $n_u = 1$. The parameters are fixed as:

$$W_1 = \begin{bmatrix} 0.80 & -0.10 & 0.40 \\ -0.50 & 0.65 & 0.20 \\ 0.25 & 0.45 & -0.55 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0.05 \\ -0.10 \\ 0.08 \end{bmatrix},$$

© 2026 the authors. Licensed under CC-BY-NC-ND.

$$W_2 = \begin{bmatrix} 0.70 & -0.25 & 0.35 \\ -0.40 & 0.50 & 0.10 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 0.02 \\ -0.01 \end{bmatrix}$$

The RNN dynamic is therefore

$$x_{k+1} = W_2 \,\mathrm{ReLU}\left(W_1 \begin{bmatrix} x \\ u \end{bmatrix} + b_1\right) + b_2.$$

Although the example considers a low-dimensional configuration ($n_x = 2$, $n_u = 1$) for visualization purposes, the proposed procedure is general and dimension-independent. Through Lemma 4 we identified eight nonempty activation regions, each corresponding to a unique ReLU activation pattern of the network. The spatial distribution of these feasible regions in the (x, u)-plane is reported in Fig. 2. For each active region, the corresponding local affine dynamics were computed exactly from the network weights and biases using the recursive propagation described in Section 3. Table 1 lists the values of A_σ, B_σ, and c_σ for all eight feasible regions. To verify the equivalence between the ReLU RNN and the derived CPWA model, both systems were simulated under identical input excitations and initial conditions. In the ReLU RNN case, the network performs a forward operation at each time step to compute the next state, which is then fed back as input to the network at the subsequent discrete instant. Conversely, for the CPWA model, the identified local affine dynamics corresponding to the active region are used to propagate the state trajectory over time. As shown in Fig. 3, the RNN and CPWA trajectories overlap perfectly throughout the horizon. The tracking error remains at the level of numerical precision (approximately 10^{-16}), confirming the exact analytical equivalence between the two formulations.

4.2 Numerical example from CPWA to ReLU *RNN*

We consider a CPWA dynamical system defined as follows

$$f_{\mathrm{PWA}}(x_k, u_k) = \begin{cases} -0.2\,x_k + 0.4\,u_k, & x_k \le 0,\ u_k \le 0, \\ 1.0\,x_k + 0.4\,u_k, & x_k > 0,\ u_k \le 0, \\ -0.2\,x_k - 0.3\,u_k, & x_k \le 0,\ u_k > 0, \\ 1.0\,x_k - 0.3\,u_k, & x_k > 0,\ u_k > 0, \end{cases}$$

This CPWA is made up of four regions, and it is continuous along the partition boundaries. Following the difference–of–convex (DC) decomposition described in Section 3.2,

Table 1. Local affine dynamics $A_\sigma, B_\sigma, c_\sigma$ for each activation pattern

Ω	Pattern σ	A_σ	B_σ	c_σ
1	(000)	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.02 \\ -0.01 \end{bmatrix}$
2	(100)	$\begin{bmatrix} 0.56 & -0.07 \\ -0.32 & 0.04 \end{bmatrix}$	$\begin{bmatrix} 0.28 \\ -0.16 \end{bmatrix}$	$\begin{bmatrix} 0.055 \\ -0.03 \end{bmatrix}$
3	(010)	$\begin{bmatrix} 0.125 & -0.1625 \\ -0.25 & 0.325 \end{bmatrix}$	$\begin{bmatrix} -0.05 \\ 0.10 \end{bmatrix}$	$\begin{bmatrix} 0.045 \\ -0.06 \end{bmatrix}$
4	(001)	$\begin{bmatrix} 0.0875 & 0.1575 \\ 0.025 & 0.045 \end{bmatrix}$	$\begin{bmatrix} -0.1925 \\ -0.055 \end{bmatrix}$	$\begin{bmatrix} 0.048 \\ -0.002 \end{bmatrix}$
5	(110)	$\begin{bmatrix} 0.685 & -0.2325 \\ -0.57 & 0.365 \end{bmatrix}$	$\begin{bmatrix} 0.23 \\ -0.06 \end{bmatrix}$	$\begin{bmatrix} 0.08 \\ -0.08 \end{bmatrix}$
6	(101)	$\begin{bmatrix} 0.6475 & -0.0875 \\ -0.295 & 0.085 \end{bmatrix}$	$\begin{bmatrix} 0.0875 \\ -0.215 \end{bmatrix}$	$\begin{bmatrix} 0.083 \\ -0.022 \end{bmatrix}$
7	(011)	$\begin{bmatrix} 0.2125 & -0.005 \\ -0.225 & 0.37 \end{bmatrix}$	$\begin{bmatrix} -0.2425 \\ -0.045 \end{bmatrix}$	$\begin{bmatrix} 0.073 \\ -0.052 \end{bmatrix}$
8	(111)	$\begin{bmatrix} 0.7725 & -0.075 \\ -0.545 & 0.41 \end{bmatrix}$	$\begin{bmatrix} 0.0375 \\ -0.115 \end{bmatrix}$	$\begin{bmatrix} 0.108 \\ -0.072 \end{bmatrix}$

the function f_{PWA} is expressed as $f_{\mathrm{PWA}} = f^+ - f^-$, where both f^+ and f^- are convex CPWA functions. From the exact convex decomposition, we obtain:

$$\begin{aligned} H^+_{1,1}&=[-0.4\ 0.025], \quad H^-_{1,1}=[-0.2\ -0.375], \quad q^\pm_{1,1} =[0], \\ H^+_{1,2}&=[+0.8\ 0.025], \quad H^-_{1,2}=[-0.2\ -0.375], \quad q^\pm_{1,2} =[0], \\ H^+_{1,3}&=[-0.4\ 0.025], \quad H^-_{1,3}=[-0.2\ +0.325], \quad q^\pm_{1,3} =[0], \\ H^+_{1,4}&=[+0.8\ 0.025], \quad H^-_{1,4}=[-0.2\ +0.325], \quad q^\pm_{1,4} =[0], \end{aligned}$$

Since $p^\pm = 2$ because H^+_1, H^-_1 have both two rows, then:

$$\begin{aligned} e^+_{1,1} &= -0.4x_k + 0.025u_k, \quad & e^-_{1,1} &= -0.2x_k - 0.375u_k, \\ e^+_{1,2} &= +0.8x_k + 0.025u_k \quad & e^-_{1,2} &= -0.2x_k - 0.375u_k, \\ e^+_{1,3} &= -0.4x_k + 0.025u_k, \quad & e^-_{1,3} &= -0.2x_k + 0.325u_k, \\ e^+_{1,4} &= +0.8x_k + 0.025u_k \quad & e^-_{1,4} &= -0.2x_k + 0.325u_k. \end{aligned}$$

Using the identity

$$\max\{a, b\} = \mathrm{ReLU}(a - b) + \mathrm{ReLU}(b) - \mathrm{ReLU}(-b),$$

the functions f^+ and f^- can be implemented by a single hidden layer of ReLU neurons, yielding:

$$f^+_{\mathrm{PWA}}(x_k, u_k) = \max\{e^+_{1,1}, e^+_{1,2}, e^+_{1,3}, e^+_{1,4}\} = \max\{e^+_{1,1}, e^+_{1,2}\},$$
$$f^-_{\mathrm{PWA}}(x_k, u_k) = \max\{e^-_{1,1}, e^-_{1,2}, e^-_{1,3}, e^-_{1,4}\} = \max\{e^-_{1,1}, e^-_{1,3}\}.$$

This yields the following equivalent network representation

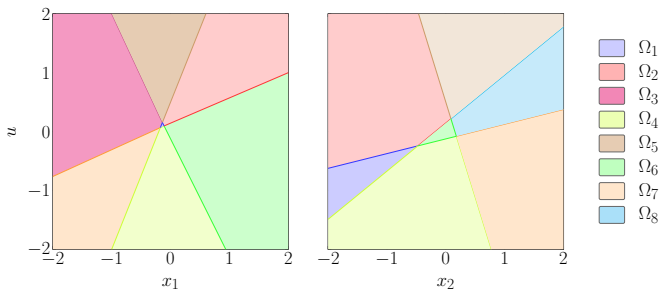


Fig. 2. Feasible regions of the ReLU RNN projected onto the planes (a) $x_2 = 0$ and (b) $x_1 = 0$. Each color corresponds to a distinct activation pattern, i.e., a locally affine subsystem (Table 1). Borders are drawn in a darker shade of the same color as their region.

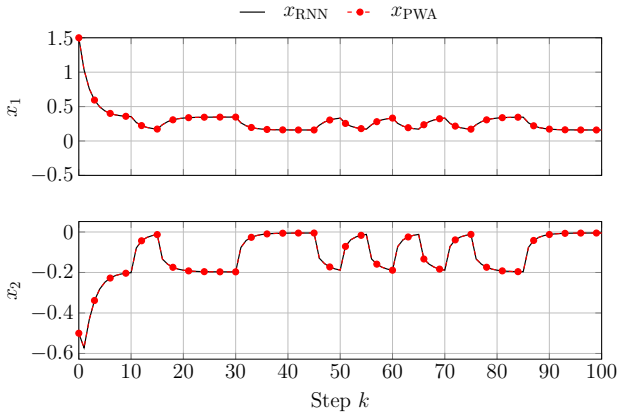


Fig. 3. Comparison between the state trajectories x_1, x_2 of RNN and CPWA dynamical system starting from the same initial condition and excited by the same input signal.

© 2026 the authors. Licensed under CC-BY-NC-ND.

$$
\begin{aligned}
f_{\text{PWA}}(x_k, u_k) &= f^{+}_{\text{PWA}}(x_k, u_k) - f^{-}_{\text{PWA}}(x_k, u_k) \\
&= \max\{e^{+}_{1,1}, e^{+}_{1,2}\} - \max\{e^{-}_{1,1}, e^{-}_{1,3}\} \\
&= +\,\text{ReLU}(e^{+}_{1,1} - e^{+}_{1,2}) + \text{ReLU}(e^{+}_{1,2}) - \text{ReLU}(-e^{+}_{1,2}) \\
&\quad - \text{ReLU}(e^{-}_{1,1} - e^{-}_{1,3}) - \text{ReLU}(e^{-}_{1,3}) + \text{ReLU}(-e^{-}_{1,3}).
\end{aligned}
$$

Therefore, the CPWA dynamical system is equivalent to the ReLU RNN with two layers $L = 2$ such that the first layer is a ReLU layer with $\alpha_1 = 0$ and with $n_1 = 6$, while the second layer is an affine layer with $\alpha_2 = 1$ and with $n_2 = 1$. The matrix weights and biases are given by

$$
W_1 = \begin{bmatrix} -1.2 & 0 \\ 0.8 & 0.025 \\ -0.8 & -0.025 \\ 0 & -0.7 \\ -0.2 & 0.325 \\ 0.2 & -0.325 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{17}
$$

$$
W_2 = [1\ 1\ -1\ -1\ -1\ 1], \quad b_2 = 0.
$$

To confirm the equivalence in the dynamic setting, the CPWA system (4.2) and its ReLU RNN realization (17) were simulated under the same excitation signal and initial condition. The two trajectories coincide at all instants, as shown in Fig. 4, with a maximum absolute error of 10^{-16} due solely to floating-point precision.

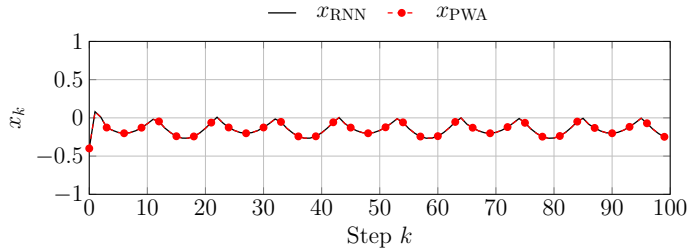


Fig. 4. Comparison between the state trajectories x_k of RNN and CPWA dynamical system starting from the same initial condition and excited by the same input signal.

5. CONCLUSIONS

This paper presented constructive procedures for translating between ReLU recurrent neural networks and discrete-time continuous piecewise-affine dynamical systems. In the RNN-to-CPWA direction, the construction computes the polyhedral activation regions and local affine dynamics directly from the network weights and biases. Conversely, a CPWA dynamical system is converted into an exact ReLU RNN realization through a difference-of-convex decomposition and max-affine representation. Numerical examples in both directions confirmed the exactness of the transformations, with trajectories matching up to machine precision. Future research will explore applications to learning-based control and model predictive design.

REFERENCES

Alur, R., Courcoubetis, C., Henzinger, T.A., and Ho, P.H. (1995). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, 209–229.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2018). Understanding deep neural networks with rectified linear units. *6th International Conference on Learning Representations, ICLR 2018*.

Bemporad, A. and Morari, M. (1999). The control of systems described by mixed logical dynamical models. *Automatica*, 35(3), 407–427.

Bemporad, A., Torrisi, F., and Morari, M. (2001). Discrete-time hybrid modeling and verification of the batch evaporator process benchmark. *European Journal of Control*, 7(4), 382–399.

Biswas, P., Grieder, P., Löfberg, J., and Morari, M. (2005). A survey on stability analysis of discrete-time piecewise affine systems. *IFAC Proceedings Volumes*, 38(1), 283–294. 16th IFAC World Congress.

Collins, P.J., Habets, L.C., Petreczky, M., and van Schuppen, J.H. (2009). Control of piecewise-affine hybrid systems - extended abstract. *IFAC Proceedings Volumes*, 42(17), 234–237. 3rd IFAC Conference on Analysis and Design of Hybrid Systems.

Darup, M.S. (2020). Exact representation of piecewise affine functions via neural networks. In *2020 European Control Conference (ECC)*, 1073–1078.

De Schutter, B. and van den Boom, T. (2004). Mpc for continuous piecewise-affine systems. *Systems& Control Letters*, 52(3), 179–192.

Dereich, S. and Kassing, S. (2022). On minimal representations of shallow relu networks. *Neural Networks*, 148, 121–128.

Ferrari-Trecate, G., Cuzzola, F.A., Mignone, D., and Morari, M. (2002). Analysis of discrete-time piecewise affine and hybrid systems. *Automatica*, 38(12), 2139–2146.

Heemels, W., De Schutter, B., and Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, 37(7), 1085–1091.

Hempel, A.B., Goulart, P.J., and Lygeros, J. (2015). Inverse parametric optimization with an application to hybrid system control. *IEEE Transactions on Automatic Control*, 60(4), 1064–1069.

Henzinger, T.A., Kopke, P.W., Puri, A., and Varaiya, P. (1998). What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1), 94–124.

Karaman, S., Sanfelice, R.G., and Frazzoli, E. (2008). Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *2008 47th IEEE Conference on Decision and Control*, 2117–2122.

Kazda, K. and Li, X. (2021). Nonconvex multivariate piecewise-linear fitting using the difference-of-convex representation. *Computers & Chemical Engineering*, 150, 107310.

Labinaz, G., Bayoumi, M.M., and Rudie, K. (1997). A survey of modeling and control of hybrid systems. *Annual Reviews in Control*, 21, 79–92.

Lazar, M. and Heemels, W. (2008). Global input-to-state stability and stabilization of discrete-time piecewise affine systems. *Nonlinear Analysis: Hybrid Systems*, 2(3), 721–734. Special Issue Section: Analysis and Design of Hybrid Systems.

Ledda, M., Deplano, D., Giua, A., and Franceschelli, M. (2025). Certification of autoencoder-based models for dynamical systems. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, 401–406.

Magnani, A. and Boyd, S.P. (2009). Convex piecewise-linear fitting. *Optimization and Engineering*, 10(1), 1–17.

Masti, D. and Bemporad, A. (2021). Learning nonlinear state–space models using autoencoders. *Automatica*, 129, 109666.

Montúfa, G., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *NeurIPS Proceedings*, volume 27.

Rodrigues, L. and Boyd, S. (2005). Piecewise-affine state feedback for piecewise-affine slab systems using convex optimization. *Systems & Control Letters*, 54(9), 835–853.

Roll, J., Bemporad, A., and Ljung, L. (2004). Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1), 37–50.

Seatzu, C., Corona, D., Giua, A., and Bemporad, A. (2006a). Optimal control of continuous-time switched affine systems. *IEEE Transactions on Automatic Control*, 51(5), 726–741.

Seatzu, C., Gromov, D., Raisch, J., Corona, D., and Giua, A. (2006b). Optimal control of discrete-time hybrid automata under safety and liveness constraints. *Nonlinear Analysis*, 65(6), 1188–1210.

© 2026 the authors. Licensed under CC-BY-NC-ND.