EXPRESSIVE POWER OF RECURRENT SPIKING NEURAL NETWORKS FOR SEQUENCE MODELING

Duc Anh Nguyen
LMU Munich
danguyen@math.lmu.de

Massimiliano Datres
LMU Munich
Munich Center for Machine Learning (MCML)
datres@math.lmu.de

Ernesto Araya
LMU Munich
Munich Center for Machine Learning (MCML)
araya@math.lmu.de

Gitta Kutyniok
LMU Munich
Munich Center for Machine Learning (MCML)
University of Tromsø
DLR-German Aerospace Center
kutyniok@math.lmu.de

ABSTRACT

Spiking neural networks (SNNs) provide a biologically inspired and potentially efficient framework for processing sequential data, but their expressive power, particularly in the dynamical setting, remains poorly understood. We establish a universality result showing that recurrent SNNs can approximate a broad class of sequence-to-sequence mappings, thereby providing theoretical support for their use in temporal learning and time series processing.

Track: Research

1 INTRODUCTION

Modern machine learning applications increasingly demand substantial computational and energy resources. While conventional deep learning models have achieved remarkable success, their high energy consumption may limit deployment in resource-constrained and real-time scenarios. Spiking neural networks (SNNs), often referred to as the third generation of artificial neural networks (ANNs) Maass (1997), have been developed as a potential energy-efficient alternative thanks to their event-driven nature inspired by biological neuronal dynamics and compatibility with neuromorphic hardware Roy et al. (2019); Mehonic et al. (2024).

Despite this potential and substantial progress in training algorithms and hardware implementations Eshraghian et al. (2023); Davies et al. (2018), the theoretical understanding of SNNs remains limited. In particular, a crucial open problem is to characterize which temporal relations between input and output sequences SNNs can represent, or equivalently, which time-dependent input–output mappings they are capable of realizing in principle.

Even within the broader ANN literature, where questions of expressivity have been extensively studied, theoretical results for sequence and time-series modeling remain relatively scarce Jiang et al. (2023). For SNNs, the corresponding theory is considerably smaller, especially in the sequential setting. This gap is particularly significant given the widespread belief that the advantages of SNNs—in terms of efficiency—manifest primarily for sparse temporal, event-based, or neuromorphic data, making a rigorous understanding of their expressive capabilities over time essential. Due to space limitations, a more detailed discussion about related works is provided in Appendix C.

In this work, we address this gap by studying the expressivity of SNNs for sequential data. We focus on discrete-time recurrent SNNs instantiated with the popular leaky integrate-and-fire (LIF) neurons, and show that such networks can represent rich classes of temporal functions, providing theoretical support for their application as both an efficient and expressive framework for sequence modeling. Our contributions can be summarized as follows.

- We prove that SNNs based on simple LIF neurons in discrete-time with recurrent connections are universal approximators for sequence-to-sequence (of finite length) functions that satisfy certain mild conditions. For target functions with binary inputs and binary outputs at every time step, there exists even an exact representation.
- We examine the benefits of recurrent connections in the investigated SNN model both theoretically and empirically.

2 PROBLEM SETUP

In general, approximation theory studies the problem of approximating elements from certain target classes using a given hypothesis set. In sequence modeling, the target functions of interest can be thought of as complex relationships between sequences. In this paper, we investigate how such functions can be approximated using SNNs in discrete time, which is formally introduced below.

2.1 SPIKING NEURAL NETWORKS IN DISCRETE TIME AS HYPOTHESIS SET

In this subsection, we introduce the formal definition of recurrent SNNs in discrete time, which is based on the notion of discrete-time SNNs in Nguyen et al. (2025). Accordingly, the key properties of the **discrete-time RLIF-SNN** model considered in this paper are described as follows:

1. **Network (spatial) architecture** $(L, \boldsymbol{n}) \in \mathbb{N} \times \mathbb{N}^{L+2}$. Neurons are arranged in layers, where L is the number of hidden layers (**depth**) and $\boldsymbol{n} = (n_0, \dots, n_{L+1})$ is the number of neurons in each layer (**width**) with input dimension $n_0 := n$ and output dimension $n_{L+1} := m$.
2. **Neuronal (temporal) dynamics.** For each hidden layer $\ell \in [L]$, the **input current** $\boldsymbol{i}^\ell(t) \in \mathbb{R}^{n_\ell}$, **spike activation** $\boldsymbol{s}^\ell(t) \in \{0,1\}^{n_\ell}$ and the **membrane potential** vector $\boldsymbol{u}^\ell(t) \in \mathbb{R}^{n_\ell}$ at time step $t \in [T]$ are given by
$$\begin{cases} \boldsymbol{i}^\ell(t) = \boldsymbol{W}^\ell \boldsymbol{s}^{\ell-1}(t) + \boldsymbol{R}^\ell \boldsymbol{s}^\ell(t-1) + \boldsymbol{b}^\ell \\ \boldsymbol{s}^\ell(t) = H\big(\beta^\ell \boldsymbol{u}^\ell(t-1) + \boldsymbol{i}^\ell(t) - \vartheta^\ell \mathbf{1}_{n_\ell}\big) \\ \boldsymbol{u}^\ell(t) = \Big(\beta^\ell \boldsymbol{u}^\ell(t-1) + \boldsymbol{i}^\ell(t) - \vartheta^\ell \mathbf{1}_{n_\ell}\Big) \odot \big(1 - \boldsymbol{s}^\ell(t)\big), \end{cases} \tag{1}$$
The last layer (corresponding to $\ell = L+1$) does not contain spike activations and follows a different dynamic
$$\begin{cases} \boldsymbol{i}^{L+1}(t) &= \boldsymbol{W}^{L+1} \boldsymbol{s}^L(t) + \boldsymbol{R}^{L+1} \boldsymbol{s}^{L+1}(t-1) + \boldsymbol{b}^{L+1}, \\ \boldsymbol{u}^{L+1}(t) &= \beta^{L+1} \boldsymbol{u}^{L+1}(t-1) + \boldsymbol{i}^{L+1}(t). \end{cases} \tag{2}$$

Here, $H : \mathbb{R} \to \{0,1\}$ denotes the Heaviside function (applied entry-wise), $\odot$ denotes the entry-wise multiplication and $\boldsymbol{s}^\ell(0) := \mathbf{0}_{n_\ell}$ (see Appendix B for a summary of notations used throughout this paper). The inputs and outputs are denoted as follows,

- **input** time series $(\boldsymbol{s}^0(t))_{t\in[T]} := \big((\boldsymbol{x}(t)\big)_{t\in[T]} \in \mathbb{R}^{n\times T}$,
- **output** time series $\big(\boldsymbol{y}(t)\big)_{t\in[T]} := \big(\boldsymbol{u}^{L+1}(t)\big)_{t\in[T]} \in \mathbb{R}^{m\times T}$.

The **(hyper)parameters** to the SNN include

- **weight matrices** $\boldsymbol{W}^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and **bias vectors** $\boldsymbol{b}^\ell \in \mathbb{R}^{n_\ell}$ for $\ell \in [L+1]$ (as commonly used in feed-forward ANNs),
- **recurrence matrices** $\boldsymbol{R}^\ell \in \mathbb{R}^{n_\ell \times n_\ell}$ for $\ell \in [L+1]$ (as in RNNs),
- **initial membrane potential** vector $\boldsymbol{u}^\ell(0) \in \mathbb{R}^{n_\ell}$, **leaky parameter** $\beta^\ell \in [0,1]$ for $\ell \in [L+1]$, and **threshold** $\vartheta^\ell \in (0,\infty)$ for $\ell \in [L]$ (which are SNN-specific temporal (hyper)parameters).

The last equation in the neuron dynamics (1) is referred to as the **reset-to-zero** mechanism of the membrane potential. Alternatively, one may replace this by the **reset-by-subtraction** rule,
$$\boldsymbol{u}^\ell(t) = \beta^\ell \boldsymbol{u}^\ell(t-1) + \boldsymbol{i}^\ell(t) - \vartheta^\ell \boldsymbol{s}^\ell(t)$$
without affecting the universality results introduced later in Section 3.

Definition 2.1 (Discrete-time RLIF-SNN). *A **recurrent spiking neural network (RSNN)** in the **discrete-time LIF model** is given by the tuple*

$$\mathbf{\Phi} := \Big(\big(\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\big)_{\ell\in[L]}, \boldsymbol{R}^\ell, \big(\boldsymbol{u}^\ell(0), \beta^\ell, \vartheta^\ell\big)_{\ell\in[L]} \Big).$$

*Given a fixed sequence length $T \in \mathbb{N}$, the RSNN Φ **realizes** the mapping $R(\mathbf{\Phi}) : \mathbb{R}^{n_{in}\times T} \to \mathbb{R}^{n_{out}\times T}$ according to the dynamics (1):*

$$R(\mathbf{\Phi})\Big(\big(\boldsymbol{x}(t)\big)_{t\in T}\Big) = \big(\boldsymbol{y}(t)\big)_{t\in[T]}.$$

In this paper, we are interested in the expressivity of RSNNs from the viewpoint of approximation theory, i.e. we want to represent or approximate some target function classes by functions realized by RSNNs. We formally define this hypothesis set as follows.

Definition 2.2 (RSNN hypothesis set). *Given input and output (spatial) dimensions $n, m \in \mathbb{N}$ and sequence length $T \in \mathbb{N}$, we denote by $\mathcal{H}^{n,m,T}_{\mathrm{RSNN}}$ the set of all functions $\mathbb{R}^{n\times T} \to \mathbb{R}^{m\times T}$ that are realizable by an RSNN. More precisely,*

$$\mathcal{H}^{n,m,T}_{\mathrm{RSNN}} := \Bigg\{ R\Bigg(\Big(\big(\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\big)_{\ell\in[L]}, \boldsymbol{R}^\ell, \big(\boldsymbol{u}^\ell(0), \beta^\ell, \vartheta^\ell\big)_{\ell\in[L]} \Big) \Bigg) \;\Bigg|\; \boldsymbol{W} \in \mathbb{R}^{n_\ell\times n_{\ell-1}}, \boldsymbol{b} \in \mathbb{R}^{n_\ell},$$
$$\boldsymbol{R}^\ell \in \mathbb{R}^{n_\ell\times n_\ell}, \boldsymbol{u}^\ell(0) \in \mathbb{R}^{n_\ell}, \beta^\ell \in [0,1], \vartheta^\ell \in \mathbb{R}_+; n_0 = n, n^{L+1} = m; L, n_1, \dots, n_L \in \mathbb{N} \Bigg\},$$

the set of all recurrent SNNs in discrete time with input and output (spatial) dimensions n and m, respectively.

2.2 Causal sequence-to-sequence mappings as target functions

By construction, the state of an RSNN at time t only relies on information from previous time steps $t' \leq t$ and cannot incorporate information from future steps $t' > t$. As a result, RSNNs cannot handle tasks where early predictions must rely on later observations. Therefore, we restrict our attention to target sequence-to-sequence functions satisfying the **causality** property, defined as follows.

Definition 2.3 (Causality). *Let $F : \mathbb{R}^{n\times T} \to \mathbb{R}^{m\times T}$, $\big(\boldsymbol{x}(t)\big)_{t\in[T]} \mapsto \big(\boldsymbol{y}(t)\big)_{t\in[T]}$ be a mapping between sequences. We say that F is **causal** if it can be described as*

$$\begin{aligned} \boldsymbol{y}(1) &= f_1\big(\boldsymbol{x}(1)\big), \\ \boldsymbol{y}(t+1) &= f_{t+1}\Big(\boldsymbol{x}(t+1), \boldsymbol{y}(t)\Big) \quad \text{for every } t \in \{1, \dots, T-1\}, \end{aligned}$$

*where $f_1 : \mathbb{R}^n \to \mathbb{R}^m$, $f_t : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ for $t \geq 2$ are arbitrary (static-static) functions. If $F : \{0,1\}^{n\times T} \to \{0,1\}^{m\times T}$ is a mapping between binary sequences and all $f_1, \dots, f_T$ are (static) binary mappings, we say that F is a **binary causal** sequence-to-sequence function.*

3 RSNN sequence approximation

In this section, we present the main result in the paper demonstrating that RSNNs are universal approximators for causal sequence-to-sequence functions with continuous transition functions.

Theorem 3.1. *(Continuous sequence-sequence function approximation) Let $n, m, T \in \mathbb{N}$, $K \subset \mathbb{R}^n$ be a compact set and let $F : K^T \to \mathbb{R}^{m\times T}$, $\big(\boldsymbol{x}(t)\big)_{t\in[T]} \mapsto \big(\boldsymbol{y}(t)\big)_{t\in[T]}$ be a causal function*

$$\begin{aligned} \boldsymbol{y}(1) &= f_1\Big(\boldsymbol{x}(1)\Big), \\ \boldsymbol{y}(t) &= f_t\Big(\boldsymbol{x}(t), \boldsymbol{y}(t-1)\Big), \quad t \geq 2, \end{aligned}$$

where $f_1 : K \to \mathbb{R}^m$ and $f_t : K \times \mathbb{R}^m \to \mathbb{R}^m$, $t \geq 2$, are continuous functions. Then for any $\epsilon > 0$, there exists a RSNN $\Psi \in \mathcal{H}^{n,m,T}_{RSNN}$ such that for every input sequence $\big(\boldsymbol{x}(t)\big)_{t\in[T]} \in K^T$ with

corresponding output sequence

$$\big(\boldsymbol{y}(t)\big)_{t\in[T]} := F\Big(\big(\boldsymbol{x}(t)\big)_{t\in[T]}\Big),$$
$$\big(\boldsymbol{z}(t)\big)_{t\in[T]} := \Psi\Big(\big(\boldsymbol{x}(t)\big)_{t\in[T]}\Big),$$

it holds

$$\sum_{t=1}^{T} \|\boldsymbol{y}(t) - \boldsymbol{z}(t)\| < \epsilon.$$

Finally, if F is binary causal, then F can be expressed exactly by an RSNN $\Psi \in \mathcal{H}^{n,m,T}_{RSNN}$.

The proof is separated into two parts: the exact binary expression is shown in Appendix D.2 and the approximation of mappings between continuous-valued sequences is presented in Appendix D.3.

4 The benefits of recurrent connections

This section aims to provide theoretical and experimental evidence demonstrating the advantages of recurrent connections. Due to space limitations, the theoretical analysis is deferred to Appendix E. For the empirical study, we consider a simple synthetic regression dataset with a total number of $m = 2048$ input-output pairs, each as a sequence of length $T \in \{4, 8\}$. More specifically, the input sequences are $\big(\boldsymbol{x}^{(i)}(t)\big)_{t\in[T]} \in \mathbb{R}^{2\times T}$, $i \in [m]$, with each coefficient at each time step uniformly drawn in $[-1, 1]$, i.e. $\boldsymbol{x}^{(i)}(t) \sim \mathrm{Unif}[-1, 1]$, and the target output sequences are $\big(y^{(i)}(t)\big)_{t\in[T]} \in \mathbb{R}^{1\times T}$, defined by $y^{(i)}(t) := \sum_{t=1}^{T} x^{(i)}_{p(t)}(t)$ where $p(t) = 1$ if t is odd and $p(t) = 2$ if t is even. This is a simple special case of causal target functions from Subsection 2.2, where in each time step t, the current label value $y^{(i)}(t)$ is increased from the previous value $y^{(i)}(t-1)$ by an amount equal to one of the two entries of $\boldsymbol{x}^{(i)}(t)$, where the exact coordinate $p(t)$ is determined by whether t is odd or even.

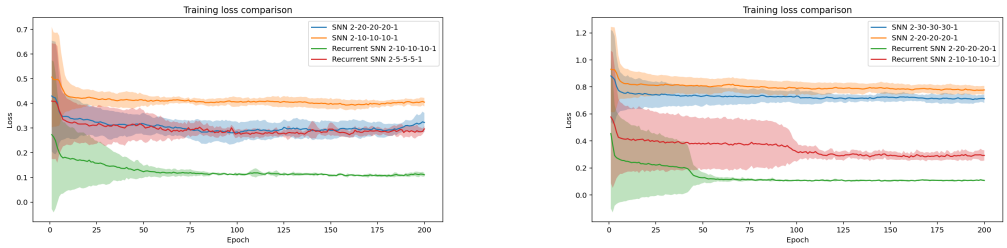
We employ `spikingjelly` Fang et al. (2023) to implement plain SNNs (without recurrent connections) and RSNNs of various sizes and train them for 200 epochs using the Adam optimizer with a learning rate of 0.01 to minimize the MSE loss, which is sufficient for all models to converge.

One can observe that in both cases $T \in \{4, 8\}$, the RSNN with the larger size (green curve) achieves the smallest training loss among all architectures. In particular, it fits the training data better than the plain SNN of the same size (orange curve). However, this comparison may not be fair as RSNNs have more trainable parameters than SNNs if they share the layer widths. Therefore, we also included a SNN of larger size (blue curve) and an RSNN of smaller size (red curve). In fact, both of them result in better alignment with the training data than the small SNN architecture while having fewer parameters than the large RSNN architecture. While having fewer parameters than both of the plain SNNs, the smaller RSNN (red curve) achieves either comparable training loss to the large SNN (when $T = 4$) or even significantly smaller training loss in comparison to both SNNs (when $T = 8$).

This suggests that including recurrent connections might be more beneficial to the expressivity of an SNN than enlarging its width while keeping the number of trainable parameters comparable, especially in the setting where T gets larger (which makes it more difficult to fit the training data).

5 Conclusion

In this work, we studied the expressive power of recurrent spiking neural networks (RSNNs) for sequential data. We established a universality result showing that RSNNs can approximate a broad family of sequence-to-sequence mappings, thereby providing a theoretical foundation for their use in temporal learning tasks. Future work includes developing more efficient constructions that reduce network size while possibly encouraging sparse spiking activity, deriving quantitative approximation bounds, and studying more complex dynamics capable of handling richer sequential dependencies.



(a) (b)

Figure 1: The learning curves of SNNs and RSNNs of various sizes, trained on a simple synthetic dataset and averaged over 5 runs. **(a)** $T = 4$. The plain SNNs have 261 and 921 trainable parameters, the RSNNs have 231 and 861 trainable parameters. **(b)** $T = 8$. The plain SNNs have 921 and 1981 trainable parameters, the RSNNs have 861 and 3321 trainable parameters.

ACKNOWLEDGMENTS

The authors acknowledge support by the project "Next Generation AI Computing (gAIn)," funded by the Bavarian Ministry of Science and the Arts and the Saxon Ministry for Science, Culture, and Tourism, as well as by the Hightech Agenda Bavaria.

Additionally, Ernesto Araya, Massimiliano Datres and Gitta Kutyniok acknowledge support by the Munich Center for Machine Learning (MCML).

Gitta Kutyniok furthermore acknowledges support by the German Research Foundation under Grants DFG-SPP-2298, KU 1446/31-1 and KU 1446/32-1, and by the Bavarian Ministry for Digital Affairs.

REFERENCES

Martin Anthony. *Discrete Mathematics of Neural Networks*. Society for Industrial and Applied Mathematics, 2001.

Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons. Theoretical foundations of deep selective state-space models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2641–2651, 2021.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

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

Shayan Hundrieser, Philipp Tuchel, Insung Kong, and Johannes Schmidt-Hieber. On the universal representation property of spiking neural networks. *arXiv:2512.16872*, 2025.

Haotian Jiang, Zhong Li, and Qianxiao Li. Approximation theory of convolutional architectures for time series modelling. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 4961–4970, 2021.

Haotian Jiang, Qianxiao Li, Zhong Li, and Shida Wang. A brief survey on the approximation theory for sequence modelling. *Journal of Machine Learning*, 2(1):1–30, 2023.

Sammy Khalife, Hongyu Cheng, and Amitabh Basu. Neural networks with linear threshold activations: structure and algorithms. *Mathematical Programming*, 206(1):333–356, 2024.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

Adnan Mehonic, Daniele Ielmini, Kaushik Roy, Onur Mutlu, Shahar Kvatinsky, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, Sabina Spiga, Sergey Savel'ev, Alexander G. Balanov, Nitin Chawla, Giuseppe Desoli, Gerardo Malavena, Christian Monzio Compagnoni, Zhongrui Wang, J. Joshua Yang, Syed Ghazi Sarwat, Abu Sebastian, Thomas Mikolajick, Stefan Slesazeck, Beatriz Noheda, Bernard Dieny, Tuo-Hung (Alex) Hou, Akhil Varri, Frank Brückerhoff-Plückelmann, Wolfram Pernice, Xixiang Zhang, Sebastian Pazos, Mario Lanza, Stefan Wiefels, Regina Dittmann, Wing H. Ng, Mark Buckwell, Horatio R. J. Cox, Daniel J. Mannion, Anthony J. Kenyon, Yingming Lu, Yuchao Yang, Damien Querlioz, Louis Hutin, Elisa Vianello, Sayeed Shafayet Chowdhury, Piergiulio Mannocci, Yimao Cai, Zhong Sun, Giacomo Pedretti, John Paul Strachan, Dmitri Strukov, Manuel Le Gallo, Stefano Ambrogio, Ilia Valov, and Rainer Waser. Roadmap to neuromorphic computing with emerging technologies. *APL Materials*, 12(10): 109201, 2024.

Duc Anh Nguyen, Ernesto Araya, Adalbert Fono, and Gitta Kutyniok. Time to spike? Understanding the representational power of spiking neural networks in discrete time. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 45954–45987, 2025.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Anton Maximilian Schäfer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. In *Artificial Neural Networks – ICANN 2006*, pp. 632–640. Springer Berlin Heidelberg, 2006.

Manjot Singh, Adalbert Fono, and Gitta Kutyniok. Expressivity of spiking neural networks through the spike response model. In *UniReps: the First Workshop on Unifying Representations in Neural Models*, 2023.

Eduardo D. Sontag. Neural nets as systems models and controllers. In *Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 73–79, 1992.

Mingze Wang and Weinan E. Understanding the expressive power and mechanisms of transformer for sequence modeling. In *Proceedings of the 38th International Conference on Neural Information Processing Systems (NeuRIPS)*, 2024.

Shida Wang and Beichen Xue. State-space models with layer-wise nonlinearity are universal approximators with exponential decaying memory. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Mikail Yayla, Mario Günzel, Burim Ramosaj, and Jian-Jia Chen. Universal approximation theorems of fully connected binarized neural networks. *arXiv:2102.02631*, 2021.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *8th International Conference on Learning Representations, ICLR*, 2020.

A ROADMAP TO THE APPENDIX

The Appendix is organized as follows

- Section B introduces the notations used throughout the paper.
- Section C provides an expanded review of related literature and an in-depth discussion of existing research.
- Section D presents the proof of Theorem 3.1. In particular, Subsection D.1 provides a few technical lemmas that are useful for the proof, Subsection D.2 proves the last statement involving binary causal target functions, and Subsection D.3 provides the rest of the proof, i.e. the part about causal target functions with continuous transition functions.
- Section E provides theoretical insights into the benefits of recurrent connections. In particular, Subsection E.1 introduces a theoretical result that compares shallow plain SNNs (without recurrent connections) and RSNNs, with the proof being presented in Subsection E.2 and E.3. Finally, Subsection E.4 introduces additional experimental results where the setting is modified from the ones shown in Section 4 in the main paper.

B NOTATIONS

Table 1 summarizes the notation used throughout the paper and appendix to aid reader comprehension. $\mathbf{1}_n$ and $\mathbf{0}_n$ denote the vectors $[1, \dots, 1]^\top$ and $[0, \dots, 0]^\top$ in $\mathbb{R}^n$, while $\mathbf{0}_{n\times m}$ denotes the all zero

Type	Notion	Notation	Domain
Network sizes	Number of spike layers	L	$\mathbb{N}$
	Layer width	n_ℓ	$\mathbb{N}$
Neuron states	Membrane potential vector	$\boldsymbol{u}^\ell(t)$	$\mathbb{R}^{n_\ell}$
	Spike activation vector	$\boldsymbol{s}^\ell(t)$	$\{0,1\}^{n_\ell}$
(Hyper-)Parameters	Weight matrix	$\boldsymbol{W}^\ell$	$\mathbb{R}^{n_\ell \times n_{\ell-1}}$
	Bias vector	$\boldsymbol{b}^\ell$	$\mathbb{R}^{n_\ell}$
	Initial membrane potential vector	$\boldsymbol{u}^\ell(0)$	$\mathbb{R}^{n_\ell}$
	Leaky parameter	β^ℓ	$[0,1]$
	Threshold value	ϑ^ℓ	$\mathbb{R}_+$
Input/output/target vectors	Latency (sequential length)	T	$\mathbb{N}$
	Input sequence	$\big(\boldsymbol{x}(t)\big)_{t\in[T]}$	$\mathbb{R}^{n_{\text{in}}\times T}$
	Output sequence	$\big(\boldsymbol{y}(t)\big)_{t\in[T]}$	$\mathbb{R}^{n_{\text{out}}\times T}$

Table 1: Table of SNN notions

matrix in $\mathbb{R}^{n\times m}$. Furthermore, we use $\|\cdot\|$ to denote the Euclidean norm, but it is worthwhile to note that in our setting, due to finiteness of both spatial and temporal dimensions, all norms involved in the same vector space are equivalent.

C RELATED WORKS

Approximation theory for sequence modeling. Despite the rapid progress of sequence models in practice, the theory of data-driven sequence modeling is still in its infancy. Early works such as Schäfer & Zimmermann (2006); Sontag (1992) primarily investigate shallow recurrent neural networks (RNNs), demonstrating their capability to approximate dynamical systems by invoking classical universal approximation theorems (UATs) for shallow ANNs Cybenko (1989); Hornik et al. (1989); Leshno et al. (1993). Our work extends such approximation results to the setting of binary activations, where the UAT-based argument breaks down due to the discontinuity of the Heaviside nonlinearity Khalife et al. (2024); Yayla et al. (2021). More recently, approximation results have been extended to more complex sequence-modeling architectures and to incorporate approximation rates, rather than focusing solely on universality; see e.g. Jiang et al. (2021); Yun et al. (2020); Wang & E. (2024); Wang & Xue (2023); Cirone et al. (2024); Jiang et al. (2023).

Expressivity of SNNs. While expressivity of ANNs has been extensively studied in the deep learning literature, analogous analyses for SNNs remain relatively scarce. Existing works, such as Maass (1997); Singh et al. (2023), often examine continuous-time SNN models, which typically rely on temporal coding schemes and the spike response model with continuous membrane potential dynamics. Another line of research Nguyen et al. (2025) investigates discrete-time SNN formulations, which are more closely aligned with commonly used implementation frameworks (e.g., Fang et al. (2023); Eshraghian et al. (2023)). Nevertheless, both lines of work primarily study the static setting in which inputs remain constant over time. In contrast, our work extends the framework of Nguyen et al. (2025) to a more general setting, demonstrating that recurrent SNNs are, in principle, also capable of processing sequential data. In particular, we are based on the discrete-time LIF-SNN models introduced in Nguyen et al. (2025), but having sequential inputs and outputs and including the recurrent connections, which are commonly supported most popular SNN implementations Fang et al. (2023); Eshraghian et al. (2023).

The concurrent work Hundrieser et al. (2025) studies a general setting consisting of continuous-time spike trains, which can be restricted to a regular grid to obtain the discrete-time setting. They show that SNNs (with slightly different neuron dynamics and no recurrent connections) are capable of expressing any target functions between spike trains that are causal, input-dominated, and satisfy the monotone scaling property. In contrast to Theorem D.7, this result achieves smaller network sizes, but their constructive proof requires the target functions to be input-dominated, i.e. the output spikes must lie within the union of input spikes, which significantly reduces the target function class. Meanwhile, our constructive proof of Theorem D.7 can be easily extended to more general notions of causality. For instance, the transition function $f_t\big(\boldsymbol{y}(t-1), \boldsymbol{x}(t)\big)$ in Def. 2.3 can be replaced by $g_t\big(\boldsymbol{x}(1), \dots, \boldsymbol{x}(t)\big)$, which is comparable to the causality notion defined in Hundrieser et al. (2025). Finally, our main result, Theorem 3.1, does not focus only on a universal representation of binary-valued sequences but also on a universal approximation of real-vector-valued sequences, which allows more complex sequence-to-sequence relations.

D Proof of results from Section 3

In this section, we introduces the proof of Theorem 3.1. We start with some preliminary results in Subsection D.1 and then investigate the case of binary causal target mappings in Subsection D.2. Based on the developed representation, the approximation of continuous-valued causal functions using RSNNs is introduced in Subsection D.3.

D.1 Auxiliary results

We first introduce several useful results about approximating or expressing static-static target functions with binary neural networks (BNNs) and then realize a few simple operations between sequences using RSNNs.

D.1.1 Static-static target functions

For the target class of continuous functions with time-independent inputs and outputs, it is well-known that BNNs (or more precisely, feed-forward ANNs with Heaviside activation function) form a dense sub-class, see e.g. Khalife et al. (2024); Nguyen et al. (2025). For convenice, we state this result in a modified manner as below.

Lemma D.1 (Static-static function approximation with BNNs)**.** *Let $n, m \in \mathbb{N}$, $K \subset \mathbb{R}^n$ be a compact set and $f : K \to \mathbb{R}^m$ be a continuous function. Then, for any $\epsilon > 0$, there exists a 2-hidden-layer binary NN $\Phi : \mathbb{R}^n \to \mathbb{R}^m$,*

$$\Phi(\boldsymbol{x}) = \boldsymbol{V} H\Big(\boldsymbol{U} H(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}_{\boldsymbol{W}}) + \boldsymbol{b}_{\boldsymbol{U}}\Big),$$

such that

$$\|f(x) - \Phi(x)\| < \epsilon \quad \forall x \in K.$$

Here, $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n}$, $\boldsymbol{U} \in \mathbb{R}^{n_2 \times n_1}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times n_2}$ are the weight matrices of the first, second (hidden) and output layer respectively, $\boldsymbol{b}_{\boldsymbol{W}} \in \mathbb{R}^{n_1}$ and $\boldsymbol{b}_{\boldsymbol{U}} \in \mathbb{R}^{n_2}$ are the bias vector in the first and second hidden layer, $n_1, n_2 \in \mathbb{N}$.

Below we state a proof sketch to this result, see also Khalife et al. (2024); Nguyen et al. (2025) for more details.

Proof sketch. Without loss of generality consider $m = 1$. Then any continuous function $f : \mathbb{R}^n \to \mathbb{R}$ can be arbitrarily well approximated by a piecewise constant function, where the regions (pieces) are polyhedra, say $P_1, \dots, P_k$. On the other hand, the indicator function $\mathbb{1}_P$ of any polyhedron $P \subseteq \mathbb{R}^n$ can be represented by a two-layer BNN (without the linear output layer). Finally, assigning function values $f_i \in \mathbb{R}^n$ to each reagion P_i and taking the weighted sum $\sum_{i=1}^k f_i \mathbb{1}_{P_i}$ yields the desired BNN approximation to f. □

In case the target function $f : \{0,1\}^n \to \{0,1\}^m$ is a binary mapping, one could also show that it can be expressed exactly by a two-layer RSNN without a linear output layer (i.e. $\boldsymbol{V} = I_m$). This is a classic result (see e.g. Anthony (2001)) and for convenience, we present the result in Lemma D.2 below.

Lemma D.2 (Binary mapping representation with BNNs)**.** *Let $n, m \in \mathbb{N}$ and $f : \{0,1\}^n \to \{0,1\}^m$ be a (binary) mapping. Then there exist $n_1 \in \mathbb{N}$, weight matrices $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n}$, $\boldsymbol{U} \in \mathbb{R}^{m \times n_1}$ and bias vectors $\boldsymbol{b_W} \in \mathbb{R}^{n_1}$, $\boldsymbol{b_U} \in \mathbb{R}^m$ such that*

$$f(\boldsymbol{x}) = H\Big(\boldsymbol{U}H(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b_W}) + \boldsymbol{b_U}\Big) \quad \forall \boldsymbol{x} \in \{0,1\}^n,$$

i.e. f can be expressed by a two-layer BNN (without a linear output layer).

One could observe that the network $\Phi = \boldsymbol{V}\Big(H\big(\boldsymbol{U}H(\boldsymbol{W}(\cdot) + \boldsymbol{b_W})\big) + \boldsymbol{b_U}\Big)$ used for the approximation in Lemma D.1 can be divided into two components:

- The function $h := H\big(\boldsymbol{U}H(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b_W}) + \boldsymbol{b_U}\big)$ (with binary outputs) stores information about the input partitioning in $\mathbb{R}^n$ as a binary vector. In other words, given an input $\boldsymbol{x} \in \mathbb{R}^n$, the output $h(\boldsymbol{x})$ indicates the region to which $\boldsymbol{x}$ belongs.
- The linear mapping $\boldsymbol{V}$ plays the role of assigning a function value to each region.

Since SNNs only have binary activations, it might be more reasonable, at a hidden layer, to store only the binary component h of a $\mathbb{R}^m$-valued function f instead of f itself. For convenience, we will define h as an approximate binary region representation of f.

Definition D.3 (Binary region representations)**.** *Let $n, m \in \mathbb{N}$, $K \subset \mathbb{R}^n$ be compact and $f : K \to \mathbb{R}^m$ be a continuous function. Given $\epsilon > 0$, let $n_1, n_2 \in \mathbb{N}$ and $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n}$, $\boldsymbol{U} \in \mathbb{R}^{n_2 \times n_1}$, $\boldsymbol{V} \in \mathbb{R}^{m \times n_2}$, $\boldsymbol{b_W} \in \mathbb{R}^{n_1}$, $\boldsymbol{b_U} \in \mathbb{R}^{n_2}$ be such that*

$$\Big\| f(\boldsymbol{x}) - \boldsymbol{V}\Big(H\big(\boldsymbol{U}H(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b_W}) + \boldsymbol{b_U}\big)\Big)\Big\| < \epsilon \quad \forall x \in K.$$

*Then, we call the function $H\big(\boldsymbol{U}H(\boldsymbol{W}(\cdot) + \boldsymbol{b_W}) + \boldsymbol{b_U}\big) : \mathbb{R}^n \to \{0,1\}^{n_2}$ an ϵ-**binary region representation** of f.*

Although Lemma D.1 shows the existence of a two-layer BNN that approximates a given continuous function, it requires that the BNN takes exact inputs and not just an approximate version. When we use SNNs to approximate target sequence-sequence functions with real-valued outputs at each time step, it is unlikely that one can save these outputs precisely without any approximation. Since the outputs of the current time step will also become the inputs to the next time step, one would need to feed approximated inputs to the next time step. While this can be done by taking the inputs as a weighted sum of the binary region representations, we still need to deal with the error estimate, which is shown in the following lemma.

Lemma D.4 (Approximated inputs)**.** *Let $n, m \in \mathbb{N}$, $K \subseteq \mathbb{R}^n$ be compact and $f : K \to \mathbb{R}^m$ be a continuous function. Let $\mathcal{H}$ be a class of functions $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ such that for any $\epsilon > 0$ there exists $\Phi_\epsilon \in \mathcal{H}$ with*

$$\|\Phi_\epsilon(x) - f(x)\| < \epsilon$$

for every $x \in K$. Fix an arbitrary $\epsilon > 0$, then there exist $\delta > 0$ and a function $\Phi \in \mathcal{H}$ such that

$$\|\Phi(y) - f(x)\| < \epsilon$$

for all $x, y \in K$ with $\|x - y\| < \delta$.

Proof. Since K is compact and f is continuous, f is also uniformly continuous. Hence, there exists $\delta > 0$ (depending on the given ϵ) such that

$$\|f(x) - f(y)\| < \frac{\epsilon}{2}$$

for any $x, y \in K$ with $\|x - y\| < \delta$. Let $\Phi \in \mathcal{H}$ be such that

$$\|\Phi(x) - f(x)\| < \frac{\epsilon}{2}$$

for every $x \in K$. By the triangle inequality, for all $x, y \in K$ with $\|x - y\| \leq \delta$ it holds

$$\|\Phi(y) - f(x)\| \leq \|\Phi(y) - f(y)\| + \|f(y) - f(x)\| < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

□

D.1.2 BASIC RSNN OPERATIONS

Here we show how a few basic binary sequence-to-sequence functions can be expressed using RSNNs, which are useful for the constructions in Appendix D.2 and D.3.

Lemma D.5 (Realizing identity mapping)**.** *Let $n, T \in \mathbb{N}$, $t_0 \in [T]$ be a fixed time step. Then the identity sequence-sequence function*

$$\mathrm{id} : \{0,1\}^{n \times T} \to \{0,1\}^{n \times T}, \big(\boldsymbol{x}(t)\big)_{t \in [T]} \mapsto \big(\boldsymbol{x}(t)\big)_{t \in [T]}$$

can be realized by an RSNN with only one single layer.

Proof. For $\boldsymbol{u}(0) := \mathbf{0}_n$, $\vartheta := 1$ and an arbitrary $\beta \in [0,1]$, we define

$$\boldsymbol{W} := \boldsymbol{I}_n \in \mathbb{R}^{n \times n}, \quad \boldsymbol{b} := \mathbf{0}_n, \quad \boldsymbol{R}^n := \mathbf{0}_{n \times n}.$$

Then, by induction, one can easily show that the output spikes $\boldsymbol{s}(t) \in \{0,1\}^n$ and the membrane potential vector $\boldsymbol{u}(t) \in \mathbb{R}^n$ satisfy $\boldsymbol{u}(t) = 0$ and thus

$$\boldsymbol{s}(t) = H\Big(\boldsymbol{x}(t) - \mathbf{1}_n\Big) = \boldsymbol{x}(t).$$

□

Lemma D.6 (Realizing extracting mapping)**.** *Let $n, T \in \mathbb{N}$, $t_0 \in [T]$ be a fixed time step and consider the following sequence-sequence function*

$$g : \{0,1\}^{n \times T} \to \{0,1\}^{n \times T}, [x(1), \ldots, x(T)] \mapsto [0, \ldots, 0, x(t_0), 0, \ldots, 0]$$

(where $x(t_0)$ is in the t_0-th position), i.e. g extracts features of the t_0-th time step and suppresses others. Then g can be realized by an RSNN, or equivalently $g \in \mathcal{H}^{n,n}_{\mathrm{RSNN}}$.

Proof. At the first layer, we define

$$\boldsymbol{W}^1 := \begin{bmatrix} \boldsymbol{I}_n \\ \mathbf{0}_{n \times n} \end{bmatrix} \in \mathbb{R}^{2n \times n}, \quad \boldsymbol{b}^1 := \begin{bmatrix} \mathbf{0}_n \\ -\mathbf{1}_n \end{bmatrix} \in \mathbb{R}^{2n}, \quad \boldsymbol{R}^1 := \mathbf{0}_{2n \times 2n}$$

with $\vartheta^1 := 1$, $\beta^1 := 1$ and $\boldsymbol{u}^1(0) := \mathbf{0}_{2n}$. Then, with any input sequence $\big(\boldsymbol{x}(t)\big)_{t \in [T]}$, it holds

$$\boldsymbol{s}^1(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \mathbf{0}_n \end{bmatrix} \in \{0,1\}^{2n},$$

i.e. the first layer realizes the identity mapping (in the first n entries) and the zero mapping (in the last n entries).

At the second layer, we define

$$\boldsymbol{W}^2 := \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0}_{n \times n} \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \quad \boldsymbol{b}^2 := \begin{bmatrix} \mathbf{0}_n \\ \frac{1}{t_0}\mathbf{1}_n \end{bmatrix} \in \mathbb{R}^{2n},$$
$$\boldsymbol{R}^2 := \begin{bmatrix} \mathbf{0}_{n \times n} & \\ & -T\boldsymbol{I}_n \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$$

with $\vartheta^2 := 1$, $\beta^2 := 1$ and $\boldsymbol{u}^2(0) = \mathbf{0}_{2n}$. Then, with the above values of $\boldsymbol{s}^1(t)$, we obtain

$$\boldsymbol{s}^2(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \delta_{t,t_0}\mathbf{1}_n \end{bmatrix} \in \{0,1\}^{2n} \quad \text{with } \delta_{t,t_0} = \begin{cases} 0, & t \neq t_0 \\ 1, & t = t_0. \end{cases}$$

Here, the first component in this layer still realizes the identity mapping (as in the first layer) while the second component (entrywise) replaces the zeros at time t_0 of the input sequence $\big(\mathbf{0}_n\big)_{t\in[T]}$ by ones. In particular, observe that the input current corresponding to the last n entries is accumulated over time and is only sufficient to surpass the threshold ϑ^2 after exactly t_0 time steps. Afterwards, the recurrent matrix $\boldsymbol{R}^2$ plays the role of resetting the voltage to a very low value so that the neurons never fire again.

Finally, at the third layer, we set

$$\boldsymbol{W}^3 := \begin{bmatrix} \frac{1}{2}\boldsymbol{I}_n & \frac{1}{2}\boldsymbol{I}_n \end{bmatrix} \in \mathbb{R}^{n\times 2n}, \quad \boldsymbol{b}^3 := \mathbf{0}_n, \quad \boldsymbol{R}^3 := \mathbf{0}_{n\times n}$$

with $\vartheta^3 := 1$, $\beta^3 := 0$ and $\boldsymbol{u}^3(0) = \mathbf{0}_n$. With this choice, the layer treats each time step separately and merges the two components of the inputs as an AND operation (denoted by $\wedge$). This means $s_i^3(t) = x_i(t) \wedge (\delta_{t,t_0})_i = \delta_{t,t_0} x_i(t)$ or equivalently

$$\boldsymbol{s}^3(t) = g\Big(\big(\boldsymbol{x}(t)\big)_{t\in[T]}\Big).$$

□

D.2 RSNN EXPRESSION OF BINARY CAUSAL FUNCTIONS

In this section, we introduce the proof of the last statement in Theorem 3.1, i.e. any binary causal sequence-to-sequence function can be realized by an RSNN. For convenience, we repeat the statement as follows.

Theorem D.7 (Binary sequence-sequence function realization)**.** *RSNNs can realize any binary causal sequence-to-sequence function. In other words, for any $n, m, T \in \mathbb{N}$, the set $\mathcal{B}^{n,m,T} := \Big\{ f : \{0,1\}^{n\times T} \to \{0,1\}^{m\times T} \;\Big|\; f \text{ is causal} \Big\}$ of all binary causal sequence-to-sequence functions with input and output (spatial) dimensions n and m and time dimension T is a subset of $\mathcal{H}^{n,m,T}_{\mathrm{RSNN}}$,*

$$\mathcal{B}^{n,m,T} \subset \mathcal{H}^{n,m,T}_{\mathrm{RSNN}}.$$

Since this is based on the fact that each of the binary functions f_t can be expressed by a two-layer BNN and thus we require more than one layer, the proof idea from earlier works on RNNs Schäfer & Zimmermann (2006), which relies on the universal approximation theorems Cybenko (1989); Hornik et al. (1989); Leshno et al. (1993) for shallow ANNs with continuous activation functions, cannot be directly applied. We modify the proof technique from Schäfer & Zimmermann (2006) by adding a network component next to the realization (or approximation) of f_t that plays the role of memorizing the inputs of the transition functions at later time steps, i.e. the inputs of $f_{t'}$ with $t' > t$.

Proof. Let $f = [f_1, \ldots, f_T] \in \mathcal{B}^{n,m} : \{0,1\}^{n\times T} \to \{0,1\}^{m\times T}$ be a causal binary sequence-sequence function. We will construct an RSNN that express f exactly, i.e. for each input sequence $\big(\boldsymbol{x}(t)\big)_{t\in[T]}$, the output sequence satisfies

$$\begin{aligned} \boldsymbol{y}(1) &= f_1\big(\boldsymbol{x}(1)\big), \\ \boldsymbol{y}(t+1) &= f_t\big(\boldsymbol{x}(t+1), \boldsymbol{y}(t)\big) \quad \text{for all } t \in [T-1]. \end{aligned}$$

To shorten the description, we set $\boldsymbol{y}(0)$ as an empty vector, so that f_1 also take an (empty) input $\boldsymbol{y}(0)$ besides $\boldsymbol{x}(1)$ and thus the above conditions become

$$\boldsymbol{y}(t) = f_t\big(\boldsymbol{x}(t), \boldsymbol{y}(t-1)\big) \quad \text{for all } t \in [T].$$

1. **BNN expression of f_t.** By the universality of BNNs (see Lemma D.2), there exist $n_t \in \mathbb{N}$ and matrices $\boldsymbol{W}^t \in \mathbb{R}^{n_t\times n}$, $\boldsymbol{U}^t \in \mathbb{R}^{m\times n_t}$, $\boldsymbol{R}^t \in \mathbb{R}^{n_t\times m}$ as well as bias vectors $\boldsymbol{b}^t_{\boldsymbol{W}} \in \mathbb{R}^{n_t}$, $\boldsymbol{b}^t_{\boldsymbol{U}} \in \mathbb{R}^m$ for $t \in [T]$ such that

$$f_t(\boldsymbol{y}, \boldsymbol{x}) = H\left(\boldsymbol{U}^t H\Big(\boldsymbol{W}^t\boldsymbol{x} + \boldsymbol{R}^t\boldsymbol{y} + \boldsymbol{b}^t_{\boldsymbol{W}}\Big) + \boldsymbol{b}^t_{\boldsymbol{U}}\right).$$

hold for all (static) binary input vectors $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{y} \in \{0,1\}^m$. Accordingly, for any input sequence $\big(\boldsymbol{x}(t)\big)_{t\in[T]}$ and corresponding output sequence $\big(\boldsymbol{y}(t)\big)_{t\in[T]} = f\Big(\big(\boldsymbol{x}(t)\big)_{t\in[T]}\Big)$ it holds

$$\boldsymbol{y}(t) = H\left(\boldsymbol{U}^t H\Big(\boldsymbol{W}^t\boldsymbol{x}(t) + \boldsymbol{R}^t\boldsymbol{y}(t-1) + \boldsymbol{b}_{\boldsymbol{W}}^t\Big) + \boldsymbol{b}_{\boldsymbol{U}}^t\right). \tag{3}$$

2. **Defining 'wrappers'.** Note that $\boldsymbol{R}^t$ is not a square matrix, which is necessary for a recurrent matrix. To overcome this problem, we set $N := \max\{n_1,\dots,n_t,m\}$ and define the following 'wrapper' matrices and vectors:

$$\tilde{\boldsymbol{y}}(t) := \begin{bmatrix}\boldsymbol{y}(t)\\ \boldsymbol{0}_{N-m}\end{bmatrix} \in \{0,1\}^N, \qquad \tilde{\boldsymbol{R}}^t := \begin{bmatrix}\boldsymbol{R}^t & \boldsymbol{0}_{n_t\times(N-m)}\\ \boldsymbol{0}_{(N-n_t)\times m} & \boldsymbol{0}_{(N-n_t)\times(N-m)}\end{bmatrix} \in \mathbb{R}^{N\times N}$$
$$\tilde{\boldsymbol{b}}_{\boldsymbol{W}}^t := \begin{bmatrix}\boldsymbol{b}_{\boldsymbol{W}}^t\\ \boldsymbol{0}_{N-n_t}\end{bmatrix} \in \mathbb{R}^N, \qquad \tilde{\boldsymbol{W}}^t := \begin{bmatrix}\boldsymbol{W}^t\\ \boldsymbol{0}_{(N-n_t)\times n}\end{bmatrix} \in \mathbb{R}^{N\times n},$$
$$\tilde{\boldsymbol{b}}_{\boldsymbol{U}}^t := \begin{bmatrix}\boldsymbol{b}_{\boldsymbol{U}}^t\\ \boldsymbol{0}\end{bmatrix} \in \mathbb{R}^N, \qquad \tilde{\boldsymbol{U}}^t := \begin{bmatrix}\boldsymbol{U}^t & \boldsymbol{0}\\ \boldsymbol{0}_{N-m} & \boldsymbol{0}\end{bmatrix} \in \mathbb{R}^{N\times N}.$$

With the introduced 'wrappers', (3) can be rewritten as

$$\tilde{\boldsymbol{y}}(t) = H\left(\tilde{\boldsymbol{U}}^t H\Big(\tilde{\boldsymbol{W}}^t\boldsymbol{x}(t) + \tilde{\boldsymbol{R}}^t\tilde{\boldsymbol{y}}(t-1) + \tilde{\boldsymbol{b}}_{\boldsymbol{W}}^t\Big) + \tilde{\boldsymbol{b}}_{\boldsymbol{U}}^t\right). \tag{4}$$

Now we define a RSNN (with the temporal parameters denoted by $(\boldsymbol{u}^\ell(0), \beta^\ell, \vartheta^\ell)$ as usual) with weight matrices $\boldsymbol{A}^\ell$, recurrence matrices $\boldsymbol{P}^\ell$ and bias vectors $\boldsymbol{b}^\ell$. In the following, $\boldsymbol{A}^\ell$, $\boldsymbol{P}^\ell$ and $\boldsymbol{b}^\ell$ (except for $\boldsymbol{A}^1$) are block matrices (or vectors) of total size $TN\times TN$ (or TN) with T^2 (or T) blocks of size $N\times N$ (or N).

3. **Realizing f_1.** We set $\beta^1 = \beta^2 := 0$, $\boldsymbol{u}^1(0) = \boldsymbol{u}^2(0) := \boldsymbol{0}_N$ and $\vartheta^1 = \vartheta^2 := 0$ (or equivalently we set it to an arbitrary value $\vartheta > 0$ and shift the bias vector by the same amount afterward). Moreover, we define the 1st layer's weight matrix $\boldsymbol{A}^1$ and bias vector $\boldsymbol{b}^1$ to be

$$\boldsymbol{A}^1 := \begin{bmatrix}\tilde{\boldsymbol{W}}^1\\ \boldsymbol{I}_N\\ \vdots\\ \boldsymbol{I}_N\end{bmatrix} \in \mathbb{R}^{TN\times N}, \quad \boldsymbol{b}^1 := \begin{bmatrix}\tilde{\boldsymbol{b}}_{\boldsymbol{W}}^1\\ \boldsymbol{0}_N\\ \vdots\\ \boldsymbol{0}_N\end{bmatrix} \in \mathbb{R}^{TN},$$

Furthermore, we define the 2nd layer's weight matrix $\boldsymbol{A}^2$, recurrence matrix $\boldsymbol{P}^2$ and bias vector $\boldsymbol{b}^2$ to be

$$\boldsymbol{A}^2 := \begin{bmatrix}\tilde{\boldsymbol{U}}^1 & & & & \\ & \tilde{\boldsymbol{W}}^2 & & & \\ & & \boldsymbol{I}_N & & \\ & & & \ddots & \\ & & & & \boldsymbol{I}_N\end{bmatrix} \in \mathbb{R}^{TN\times TN}, \quad \boldsymbol{b}^2 := \begin{bmatrix}\tilde{\boldsymbol{b}}_{\boldsymbol{U}}^1\\ \tilde{\boldsymbol{b}}_{\boldsymbol{W}}^2\\ \boldsymbol{0}_N\\ \vdots\\ \boldsymbol{0}_N\end{bmatrix} \in \mathbb{R}^{TN},$$

$$\boldsymbol{P} := \begin{bmatrix}\boldsymbol{0}_{N\times N} & \boldsymbol{0}_{N\times N} & \dots & \boldsymbol{0}_{N\times N}\\ \tilde{\boldsymbol{R}}^2 & \boldsymbol{0}_{N\times N} & \dots & \boldsymbol{0}_{N\times N}\\ \boldsymbol{0}_{N\times N} & \boldsymbol{0}_{N\times N} & \dots & \boldsymbol{0}_{N\times N}\\ \vdots_{N\times N} & \vdots_{N\times N} & \ddots & \vdots_{N\times N}\\ \boldsymbol{0}_{N\times N} & \boldsymbol{0}_{N\times N} & \dots & \boldsymbol{0}_{N\times N}\end{bmatrix} \in \mathbb{R}^{TN\times TN}$$

Then the 1st and 2nd layer activations at time step t become

$$\boldsymbol{s}^1(t) = \begin{bmatrix} H\Big(\tilde{\boldsymbol{W}}^1\boldsymbol{x}(t) + \tilde{\boldsymbol{b}}^1_{\boldsymbol{W}}\Big) \\ \boldsymbol{x}(t) \\ \vdots \\ \boldsymbol{x}(t) \end{bmatrix} \in \{0,1\}^{NT},$$

$$\boldsymbol{s}^2(t) = \begin{bmatrix} H\Big(\tilde{\boldsymbol{U}}^1 H\big(\tilde{\boldsymbol{W}}^1\boldsymbol{x}(t) + \tilde{\boldsymbol{b}}^t_{\boldsymbol{W}}\big) + \tilde{\boldsymbol{b}}^t_{\boldsymbol{U}}\Big) \\ H\big(\tilde{\boldsymbol{W}}^2\boldsymbol{x}(t) + \tilde{\boldsymbol{R}}^2\boldsymbol{s}^1(t-1) + \boldsymbol{b}^2_{\boldsymbol{W}}\big) \\ \boldsymbol{x}(t) \\ \vdots \\ \boldsymbol{x}(t) \end{bmatrix} \in \{0,1\}^{NT}$$

In particular, the first block of N entries in $\boldsymbol{s}^2(1)$ at time step $t = 1$ satisfies

$$\boldsymbol{s}^1_{[1:N]}(1) = H\Big(\tilde{\boldsymbol{U}}^1 H\big(\tilde{\boldsymbol{W}}^1\boldsymbol{x}(1) + \tilde{\boldsymbol{b}}^t_{\boldsymbol{W}}\big) + \tilde{\boldsymbol{b}}^t_{\boldsymbol{U}}\Big) = \tilde{\boldsymbol{y}}(1).$$

Furthermore, it also holds

$$\begin{cases} \boldsymbol{s}^2_{[1:N]}(1) & = \tilde{\boldsymbol{y}}(1) \\ \boldsymbol{s}^2_{[N+1:2N]}(2) & = H\Big(\tilde{\boldsymbol{W}}^2\boldsymbol{x}(2) + \tilde{\boldsymbol{R}}^2\tilde{\boldsymbol{y}}(1) + \tilde{\boldsymbol{b}}^2_{\boldsymbol{W}}\Big) \\ \boldsymbol{s}^2_{(i-1)N+1:iN}(i) & = \boldsymbol{x}(i) \quad \forall i \geq 3. \end{cases} \tag{5}$$

This means that among T blocks (of size N) of $\boldsymbol{s}^2$, the 1st block realizes $\tilde{\boldsymbol{y}}(1)$ at time $t = 1$, the 2nd block realizes $H\Big(\tilde{\boldsymbol{W}}^2\boldsymbol{x}(2) + \tilde{\boldsymbol{R}}^2\tilde{\boldsymbol{y}}(1) + \tilde{\boldsymbol{b}}^2_{\boldsymbol{W}}\Big)$ at time $t = 2$ and the subsequent blocks play the role of memorizing the inputs, i.e. the i-th block at time step i memorizes $\boldsymbol{x}(i)$ for any $i \geq 3$.

4. **Realizing f_t for $t \geq 2$.** We define the weight and recurrence matrices as well as bias vectors in the layers $t = 3, \ldots, T$ by

$$\boldsymbol{A}^t := \begin{bmatrix} \boldsymbol{I}_{(t-2)N} & & & \\ & \tilde{\boldsymbol{U}}^{t-1} & & \\ & & \tilde{\boldsymbol{W}}^t & \\ & & & \boldsymbol{I}_{(T-t)N} \end{bmatrix} \in \mathbb{R}^{TN \times TN},$$

$$\boldsymbol{P}^t := \begin{bmatrix} \boldsymbol{0}_{(t-1)N\times(t-2)N} & \boldsymbol{0}_{(t-1)N\times N} & \boldsymbol{0}_{(t-1)N\times(T-t+1)N} \\ \boldsymbol{0}_{N\times(t-2)N} & \tilde{\boldsymbol{R}}^t & \boldsymbol{0}_{N\times(T-t+1)N} \\ \boldsymbol{0}_{(T-t)N\times(t-2)N} & \boldsymbol{0}_{(T-t)N\times N} & \boldsymbol{0}_{(T-t)N\times(T-t+1)N} \end{bmatrix} \in \mathbb{R}^{TN\times TN}$$

(with $\tilde{\boldsymbol{R}}^t$ at block $(t, t-1)$, so that it is multiplied with the $(t-1)$-th block of the $\boldsymbol{s}^t$)

$$\boldsymbol{b}^t := \begin{bmatrix} \boldsymbol{0}_{(t-2)N} \\ \tilde{\boldsymbol{b}}^t_{\boldsymbol{U}} \\ \tilde{\boldsymbol{b}}^t_{\boldsymbol{W}} \\ \boldsymbol{0}_{(T-t)N} \in \mathbb{R}^{TN\times N} \end{bmatrix}$$

($\tilde{\boldsymbol{b}}^t_{\boldsymbol{U}}$ and $\tilde{\boldsymbol{b}}^t_{\boldsymbol{W}}$ are the $(t-1)$-th and t-block).

Moreover, we set $\boldsymbol{u}^t(0) := 0$, $\beta^t := 0$, $\vartheta^t := 0$. With this construction, we obtain

$$\boldsymbol{s}^t_{[(i-1)N+1:iN]}(i) = \begin{cases} \tilde{\boldsymbol{y}}(i) & \text{for } i \leq t-1, \\ H\Big(\tilde{\boldsymbol{W}}^t\boldsymbol{x}(t) + \tilde{\boldsymbol{R}}^t\tilde{\boldsymbol{y}}(t-1) + \tilde{\boldsymbol{b}}^t_{\boldsymbol{W}}\Big), & \text{for } i = t, \\ \boldsymbol{x}(i) & \text{for } i \geq t+1. \end{cases} \tag{6}$$

At layer $\ell = T + 1$, we obtain

$$\boldsymbol{s}^{T+1}_{[(i-1)N+1:iN]}(i) = \tilde{\boldsymbol{y}}(i) \in \mathbb{R}^N, \tag{7}$$

which means that the i-th block of $\boldsymbol{s}^{T+1}$ realize $\tilde{\boldsymbol{y}}(i)$ at the time step i.

5. **Extraction layers:** With the help of Lemma D.6, we define the next layers to extract $y(i)$ both spatially and temporally. For this, define the following auxiliary matrices and biases vectors (taken from Lemma D.6)

$$\boldsymbol{V}_1 := \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0}_{m\times m} \end{bmatrix} \in \mathbb{R}^{2m\times m}, \quad \boldsymbol{d}_1 := \begin{bmatrix} \boldsymbol{0}_m \\ -\boldsymbol{1}_m \end{bmatrix} \in \mathbb{R}^{2m}$$
$$\boldsymbol{V}_2 := \begin{bmatrix} \boldsymbol{I}_m & \\ & \boldsymbol{0}_{m\times m} \end{bmatrix} \in \mathbb{R}^{2m\times 2m}, \quad \boldsymbol{d}_2^{(t)} := \begin{bmatrix} \boldsymbol{0}_m & \frac{1}{t}\boldsymbol{1}_m \end{bmatrix} \in \mathbb{R}^{2m},$$
$$\boldsymbol{V}_3 := \begin{bmatrix} \frac{1}{2}\boldsymbol{I}_m & \frac{1}{2}\boldsymbol{I}_m \end{bmatrix} \in \mathbb{R}^{m\times 2m}, \quad \boldsymbol{Q}_2 := \begin{bmatrix} \boldsymbol{0}_{m\times m} & \\ & -T\boldsymbol{I}_m \end{bmatrix} \in \mathbb{R}^{2m\times 2m}.$$

We define the next layer by

$$\boldsymbol{A}^{T+2} := \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{0}_{2m\times(N-m)} & & & \\ & & \ddots & \ddots & \\ & & & \boldsymbol{V}_1 & \boldsymbol{0}_{2m\times(N-m)} \end{bmatrix} \in \mathbb{R}^{2Tm\times TN}$$

and

$$\boldsymbol{P}^{T+2} := \boldsymbol{0}_{2Tm\times 2Tm}, \quad \boldsymbol{b}^{T+2} := \begin{bmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_1 \end{bmatrix} \in \mathbb{R}^{2Tm},$$

together with $\beta^{T+2} := 1$, $\vartheta^{T+2} := 1$ and $\boldsymbol{u}^{T+2}(0) := \boldsymbol{0}_{2Tm}$.

Similarly, we define

$$\boldsymbol{A}^{T+3} := \begin{bmatrix} \boldsymbol{V}_2 & & \\ & \ddots & \\ & & \boldsymbol{V}_2 \end{bmatrix} \in \mathbb{R}^{2Tm\times 2Tm}, \quad \boldsymbol{b}^{T+3} := \begin{bmatrix} \boldsymbol{d}_2^{(1)} \\ \vdots \\ \boldsymbol{d}_2^{(T)} \end{bmatrix} \in \mathbb{R}^{2Tm},$$
$$\boldsymbol{P}^{T+3} := \begin{bmatrix} \boldsymbol{R}_2 & & \\ & \ddots & \\ & & \boldsymbol{R}_2 \end{bmatrix} \in \mathbb{R}^{2Tm},$$

with $\beta^{T+3} := 1$, $\vartheta^{T+3} := 1$ as well as $\boldsymbol{u}^{T+3}(0) := \boldsymbol{0}_{2Tm}$, and

$$\boldsymbol{A}^{T+4} := [\boldsymbol{V}_3 \dots \boldsymbol{V}_3] \in \mathbb{R}^{m\times 2Tm}, \boldsymbol{b}^{T+4} := \boldsymbol{0}_m, \boldsymbol{P}^{T+4} := \boldsymbol{0}_{m\times m},$$

together with $\beta^{T+4} := 0$, $\vartheta^{T+4} := 1$, $\boldsymbol{u}^{T+3}(0) := \boldsymbol{0}_m$.

Applying Lemma D.6 to each block of $\boldsymbol{s}^{T+1}$, we obtain that $\boldsymbol{s}^{T+3}$ contains $\boldsymbol{y}(t)$ and $\boldsymbol{1}_m$ at block t and time step t, and $\boldsymbol{0}_m$ elsewhere. Therefore, the $(T + 4)$-th layer, as defined above (in particular, it includes the summation and is not as a block diagonal matrix, because taking the sum of zeros does not change the output), gives output $\boldsymbol{s}^{T+4}(t) = \boldsymbol{y}(t)$ at time step t.

□

D.3 RSNN APPROXIMATION OF CONTINUOUS-VALUED CAUSAL FUNCTIONS

In this section, we present the rest of the proof of Theorem 3.1. In comparison to Theorem D.7, each transition function f_t is no longer a binary mapping. Thus, we can only hope to approximate

it instead of expressing it exactly. Another difficulty here is that we need to pass the approximate information we obtain through f_t as inputs to f_{t+1} in the form of binary spikes rather than in the original real-valued form. To overcome the problem of non-exact inputs, we need to assume continuity of the transition functions and apply Lemma D.4. Furthermore, the problem with extremely limited activation values can be surpassed by observing that even though binary spikes cannot encode function values, they can still store information about the binary region representation of f_t.

Theorem D.8. *(Continuous sequence-sequence function approximation) Let $n, m, T \in \mathbb{N}$, $K \subset \mathbb{R}^n$ be a compact set and let $F : K^T \to \mathbb{R}^{m\times T}$, $\big(\boldsymbol{x}(t)\big)_{t\in[T]} \mapsto \big(\boldsymbol{y}(t)\big)_{t\in[T]}$ be a causal function*

$$\begin{aligned}\boldsymbol{y}(1) &= f_1\Big(\boldsymbol{x}(1)\Big),\\ \boldsymbol{y}(t) &= f_t\Big(\boldsymbol{x}(t), \boldsymbol{y}(t-1)\Big), \quad t \geq 2,\end{aligned}$$

where $f_1 : K \to \mathbb{R}^m$ and $f_t : K \times \mathbb{R}^m \to \mathbb{R}^m$, $t \geq 2$, are continuous functions. Then for any $\epsilon > 0$, there exists a RSNN $\Psi \in \mathcal{H}^{n,m}_{RSNN}$ such that for all input sequences $\big(\boldsymbol{x}(t)\big)_{t\in[T]} \in K^T$ with corresponding outputs

$$\begin{aligned}\big(\boldsymbol{y}(t)\big)_{t\in[T]} &:= F\Big(\big(\boldsymbol{x}(t)\big)_{t\in[T]}\Big),\\ \big(\boldsymbol{z}(t)\big)_{t\in[T]} &:= \Psi\Big(\big(\boldsymbol{x}(t)\big)_{t\in[T]}\Big),\end{aligned}$$

it holds

$$\sum_{t=1}^{T} \|\boldsymbol{y}(t) - \boldsymbol{z}(t)\| < \epsilon.$$

Remark (Necessity of causality and transition continuity)**.** *The causality assumption in Theorem 3.1 can be extended to a more general notion of causality where we require $\boldsymbol{y}(t+1)$ to depend directly on inputs $\boldsymbol{x}(i)$ with $i \leq t$, i.e. $\boldsymbol{y}(t+1) = g_t\Big(\boldsymbol{x}(1), \ldots, \boldsymbol{x}(t+1)\Big)$ rather than just through the proxy $\boldsymbol{y}(t)$. However, it cannot be removed completely since RSNNs process information only forward in time. Similarly, the continuity of transition functions can be weakened, but regularity is generally not removable in the sense that it is rather impossible, even in the case $T = 1$, to approximate a highly irregular function merely with a finite number of constant pieces.*

Proof. • **Step 1: BNN approximation of f_t and their inputs.** Observe that by the continuity of f_t and the compactness of K, it follows that each $\boldsymbol{y}(t)$ lies in a compact set $Y_t \subset \mathbb{R}^m$ for every t. Hence, by Lemma D.1, for arbitrary $\epsilon_1, \ldots, \epsilon_T > 0$, there exist BNNs

$$\Phi^{(1)}(\boldsymbol{x}) = \boldsymbol{V}^1 H\left(\boldsymbol{U}^1 H\Big(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1_{\boldsymbol{W}}\Big) + \boldsymbol{b}^1_{\boldsymbol{U}}\right),$$

$$\Phi^{(t)}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{V}^t H\left(\boldsymbol{U}^t H\Big(\boldsymbol{W}^t x + \boldsymbol{R}^t y + b^t_W\Big) + b^t_U\right),\ t \geq 2$$

with weight matrices $\boldsymbol{W}^t \in \mathbb{R}^{n_t\times n}$, $\boldsymbol{R}^t \in \mathbb{R}^{n_t\times m}$, $\boldsymbol{U}^t \in \mathbb{R}^{m_t\times n_t}$, $\boldsymbol{V}^t \in \mathbb{R}^{m\times m_t}$ and biases $b^{(t)}_W \in \mathbb{R}^{n_t}$, $\boldsymbol{b}^{(t)}_U \in \mathbb{R}^{m_t}$ such that

$$\Big|f_1(\boldsymbol{x}) - \Phi^{(1)}(\boldsymbol{x})\Big| < \epsilon_1 \quad \forall \boldsymbol{x} \in K, \tag{8}$$

and for every $t \in [T] \setminus \{1\}$ as well as for all $(\boldsymbol{x}, \boldsymbol{y}), (\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \in K \times Y_t$ with

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| + \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\| < 2\epsilon_t,$$

it holds

$$\Big|f_{t+1}(\boldsymbol{x}, \boldsymbol{y}) - \Phi^{(t+1)}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})\Big| < \epsilon_{t+1}. \tag{9}$$

On the other hand, for $\delta := \min\{\epsilon_t : t \in [T]\} > 0$, there exist weight matrices $\boldsymbol{E}_1$, $\boldsymbol{E}_2$, $\boldsymbol{E}_3$ and bias vectors $\boldsymbol{d}_1$, $\boldsymbol{d}_2$ such that

$$\left\|\boldsymbol{x} - \boldsymbol{E}_3 H\left(\boldsymbol{E}_2 H\Big(\boldsymbol{E}_1\boldsymbol{x} + \boldsymbol{d}_1\Big) + \boldsymbol{d}_2\right)\right\| < \delta \quad \forall \boldsymbol{x} \in K. \tag{10}$$

- **Step 2: Define the 'wrappers'.** This step is totally similar to Step 2 in the proof of Theorem D.7 with setting $N := \max\{m, m_t, n_t : t \in \mathbb{N}\}$, adding zeros to make $\boldsymbol{R}^t$ square matrices and complementing the other matrices (as well as bias vectors) with zeros to match the new size. To simplify notations, from now on we omit the tildes and ignore the exact dimensions of the matrices. Furthermore, with $\boldsymbol{s}_i^\ell(t)$ the sub-index i means the i-th block rather than the exact i-th coordinate.

- **Step 3: Define the first three layers.** We define the weight and recurrence matrix as well as bias vector in the 1st and 2nd layers to be

$$\boldsymbol{A}^1 := \begin{bmatrix} \boldsymbol{W}^1 \\ \boldsymbol{E}_1 \\ \vdots \\ \boldsymbol{E}_1 \end{bmatrix}, \quad \boldsymbol{P}^1 := \boldsymbol{0}, \quad \boldsymbol{b}^1 = \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{W}}^1 \\ \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_1 \end{bmatrix},$$

$$\boldsymbol{A}^2 := \begin{bmatrix} \boldsymbol{U}^1 & & & \\ & \boldsymbol{E}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{E}_2 \end{bmatrix}, \boldsymbol{P}^2 := \boldsymbol{0}, \quad \boldsymbol{b}^2 = \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{U}}^1 \\ \boldsymbol{d}_2 \\ \vdots \\ \boldsymbol{d}_2 \end{bmatrix},$$

$$\boldsymbol{A}^3 := \begin{bmatrix} \boldsymbol{I} & & & & \\ & \boldsymbol{W}^2\boldsymbol{E}_3 & & & \\ & & \boldsymbol{I} & & \\ & & & \ddots & \\ & & & & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{P}^3 := \begin{bmatrix} \boldsymbol{0} & & & \\ \boldsymbol{R}^2\boldsymbol{V}^1 & \boldsymbol{0} & & \\ & & \ddots & \\ & & & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{b}^3 := \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b}_{\boldsymbol{W}}^2 \\ \boldsymbol{0} \\ \dots \\ \boldsymbol{0} \end{bmatrix}$$

Furthermore, we set $\beta^1 = \beta^2 = \beta^3 := 0$, $\boldsymbol{u}^1(0) = \boldsymbol{u}^2(0) = \boldsymbol{u}^3(0) := \boldsymbol{0}$ and $\vartheta^1 = \vartheta^2 = \vartheta^3 := 0$. Then, by definition $\boldsymbol{s}^2(t) = H\big(\boldsymbol{A}^2\boldsymbol{s}^1(t) + \boldsymbol{P}^2\boldsymbol{s}^2(t-1) + \boldsymbol{b}^2\big)$, the 2nd layer's activations are given by [1]

$$\boldsymbol{s}^2(1) = \begin{bmatrix} \boldsymbol{g}_1 \\ * \\ \vdots \\ * \end{bmatrix}, \quad \boldsymbol{s}^2(t) = \begin{bmatrix} * \\ * \\ \vdots \\ \boldsymbol{z}_t \\ \vdots \\ * \end{bmatrix} \quad \text{for } t \geq 2.$$

Here, $\boldsymbol{g}_1$ is the 1st block of $\boldsymbol{s}^2(1)$,

$$\boldsymbol{g}_1 := \boldsymbol{s}_1^2(1) = H\left(\boldsymbol{U}^1 H\Big(\boldsymbol{W}^1\boldsymbol{x}(1) + \boldsymbol{b}_{\boldsymbol{W}}^1\Big) + \boldsymbol{b}_{\boldsymbol{U}}^1\right),$$

which satisfies

$$\big\|\boldsymbol{V}^1\boldsymbol{g}_1 - \boldsymbol{y}(1)\big\| = \big\|\boldsymbol{V}^1\boldsymbol{g}_1 - f_1\big(\boldsymbol{x}(1)\big)\big\| < \epsilon_1, \tag{11}$$

while $\boldsymbol{z}_t$ is the t-th block of $\boldsymbol{s}^2(t)$ with $t \geq 2$,

$$\boldsymbol{z}_t := \boldsymbol{s}_t^2(t) := H\left(\boldsymbol{E}_2 H\Big(\boldsymbol{E}_1\boldsymbol{x}(t) + \boldsymbol{d}^t\Big) + \boldsymbol{d}_2\right), \tag{12}$$

which satisfies

$$\|\boldsymbol{E}_3\boldsymbol{z}_t - \boldsymbol{x}(t)\| < \delta \quad \text{for all } t \geq 2. \tag{13}$$

[1] Here $*$ denotes an expression that one could write explicitly, but is not relevant further in the proof and hence omitted for convenience.

By definition $\boldsymbol{s}^3(t) = H\big(\boldsymbol{A}^3\boldsymbol{s}^2(t) + \boldsymbol{P}^3\boldsymbol{s}^3(t-1) + \boldsymbol{b}^3\big)$, we obtain

$$\boldsymbol{s}^3(1) = \begin{bmatrix} \boldsymbol{g}_1 \\ * \\ \vdots \\ * \end{bmatrix},\ \boldsymbol{s}^3(2) = \begin{bmatrix} * \\ H\Big(\boldsymbol{W}^2\boldsymbol{E}_3\boldsymbol{z}_2 + \boldsymbol{R}^2\boldsymbol{V}^1\boldsymbol{g}_1 + \boldsymbol{b}^2_{\boldsymbol{W}}\Big) \\ * \\ \vdots \\ * \end{bmatrix},\ \boldsymbol{s}^3(t) = \begin{bmatrix} * \\ \vdots \\ \boldsymbol{z}_t \\ \vdots \\ * \end{bmatrix} \text{ for } t \geq 3.$$

- **Step 4: Define approximating layers.** For layer $t+1$ with $t \in \{3, \ldots, T\}$, we define

$$\boldsymbol{A}^{t+1} := \begin{bmatrix} \boldsymbol{I} & & & & & & & \\ & \ddots & & & & & & \\ & & \boldsymbol{I} & & & & & \\ & & & \boldsymbol{U}^{t-1} & & & & \\ & & & & \boldsymbol{W}^t\boldsymbol{E}_3 & & & \\ & & & & & \boldsymbol{I} & & \\ & & & & & & \ddots & \\ & & & & & & & \boldsymbol{I} \end{bmatrix}$$

with $\boldsymbol{U}^{t-1}$ at block $(t-1) \times (t-1)$ and $\boldsymbol{W}^t\boldsymbol{E}_3$ at block $t \times t$,

$$\boldsymbol{P}^{t+1} := \begin{bmatrix} \boldsymbol{0} & & & & & \\ & \ddots & & & & \\ & & \boldsymbol{0} & & & \\ & & \boldsymbol{R}^t\boldsymbol{V}^{t-1} & \boldsymbol{0} & & \\ & & & & \ddots & \\ & & & & & \boldsymbol{0} \end{bmatrix}$$

with $\boldsymbol{R}^t\boldsymbol{V}^{t-1}$ at block $t \times (t-1)$,

$$\boldsymbol{b}^{t+1} := \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{b}^{t-1}_{\boldsymbol{U}} \\ \boldsymbol{b}^t_{\boldsymbol{W}} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}$$

with $\boldsymbol{b}^t_{\boldsymbol{U}}$ at block $(t-1)$ and $\boldsymbol{b}^t_{\boldsymbol{W}}$ at block t. The final layer, layer $T+2$ (corresponding to $t = T+1$) in this class of layers is defined by

$$\boldsymbol{A}^{T+2} := \begin{bmatrix} \boldsymbol{I} & & & \\ & \ddots & & \\ & & \boldsymbol{I} & \\ & & & \boldsymbol{U}^T \end{bmatrix}, \quad \boldsymbol{P}^{T+2} := \boldsymbol{0}, \quad \boldsymbol{b}^{T+2} := [\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{b}^T_{\boldsymbol{U}}]^\top.$$

For the temporal parameters, set $\beta^{t+1} := 0$, $\boldsymbol{u}^{t+1}(0) := \boldsymbol{0}$, $\vartheta^{t+1} := 0$ for all $t \in \{3, \ldots, T+1\}$.

Lemma D.9. *With this choice, the following statements hold:*

1. *The activations of layer $t+1$, for $t \in \{3, \ldots, T\}$, satisfy*

$$\boldsymbol{s}_i^{t+1}(i) = \begin{cases} \boldsymbol{g}_i & \text{for } i \in \{1, \ldots, t-1\} \\ \boldsymbol{z}_i & \text{for } i \in \{t+1, \ldots, T\} \\ H\Big(\boldsymbol{W}^t\boldsymbol{E}_3\boldsymbol{z}_t + \boldsymbol{R}^t\boldsymbol{V}^{t-1}\boldsymbol{g}_{t-1} + \boldsymbol{b}^t_{\boldsymbol{W}}\Big) & \text{for } i = t, \end{cases}$$

and $\boldsymbol{s}_i^{t+2}(i) = \boldsymbol{g}_i$ *for all* $i \in [T]$*. Here,* $\boldsymbol{z}_t$ *is defined as in (12) and* $\boldsymbol{g}_i$ *is defined recursively by*

$$\boldsymbol{g}_i := H\left(\boldsymbol{U}^i H\Big(\boldsymbol{W}^i \boldsymbol{E}_3 \boldsymbol{z}_i + \boldsymbol{R}^i \boldsymbol{V}^{i-1} \boldsymbol{g}_{i-1} + \boldsymbol{b}_{\boldsymbol{W}}^i\Big) + \boldsymbol{b}_U^i\right)$$

for $i \in \{2, \ldots, T\}$.

2. *For every* $t \in [T]$ *it holds*

$$\left\|\boldsymbol{V}^t \boldsymbol{g}_t - \boldsymbol{y}(t)\right\| < \epsilon_t$$

Proof. 1. One can rewrite the desired statement as

$$\boldsymbol{s}^{t+1}(i) = \begin{bmatrix} * \\ \vdots \\ * \\ \boldsymbol{g}_i \\ * \\ \vdots \\ * \end{bmatrix} \text{ for } i \leq t-1, \quad \boldsymbol{s}^{t+1}(i) = \begin{bmatrix} * \\ \vdots \\ * \\ \boldsymbol{z}_i \\ * \\ \vdots \\ * \end{bmatrix} \text{ for } i \geq t+1,$$

$$\boldsymbol{s}^{t+1}(t) = \begin{bmatrix} * \\ \vdots \\ * \\ H\Big(\boldsymbol{W}^t \boldsymbol{E}_3 \boldsymbol{z}_t + \boldsymbol{R}^t \boldsymbol{V}^{t-1} \boldsymbol{g}_{t-1} + \boldsymbol{b}_{\boldsymbol{W}}^t\Big) \\ * \\ \vdots \\ * \end{bmatrix}$$

where, in each vector, the relevant block (i.e. not marked with $*$) is equal to the corresponding time step. We prove the claim by induction. Accordingly, consider the transformation from $\boldsymbol{s}^t$ to $\boldsymbol{s}^{t+1}$, namely by definition

$$\boldsymbol{s}^{t+1}(i) = H\Big(\boldsymbol{A}^t \boldsymbol{s}^t(i) + \boldsymbol{P}^t \boldsymbol{s}^{t+1}(i-1) + \boldsymbol{b}^t\Big).$$

Since the first $(t-2)$ blocks in $\boldsymbol{A}^{t+1}$, $\boldsymbol{P}^{t+1}$ and $\boldsymbol{b}^{t+1}$ simply realize the identity, it follows that for all $i \leq t-2$, the i-th block of $\boldsymbol{s}^{t+1}(i)$ and that of $\boldsymbol{s}^t(i)$ are the same and equal to $\boldsymbol{g}_i$ (by induction). Similarly, the last $T-t-1$ blocks (i.e. from $t+2$ to T) in $\boldsymbol{A}^{t+1}$, $\boldsymbol{P}^{t+1}$ and $\boldsymbol{b}^{t+1}$ realize the identity mapping, which implies that those corresponding blocks in $\boldsymbol{s}^{t+1}(i)$ for $i \geq t+1$ are equal to those in $\boldsymbol{s}^t(i)$ and thus equal to $\boldsymbol{z}_i$. It remains to consider the two time steps where the actual transformation happens, i.e. $i \in \{t-1, t\}$.
For $i = t-1$, it holds

$$\begin{aligned} \boldsymbol{s}_{t-1}^{t+1}(t-1) &= H\left(\boldsymbol{U}^{t-1} \boldsymbol{s}_{t-1}^t(t-1) + \boldsymbol{b}_U^{t-1}\right) \\ &\overset{\text{ind.}}{=} H\left(\boldsymbol{U}^{t-1} H\Big(\boldsymbol{W}^{t-1} \boldsymbol{E}_3 \boldsymbol{z}_{t-1} + \boldsymbol{R}^{t-1} \boldsymbol{V}^{t-2} \boldsymbol{g}_{t-2}\Big) + \boldsymbol{b}_U^{t-1}\right) \\ &= \boldsymbol{g}_{t-1}. \end{aligned}$$

For $i = t$, it holds

$$\begin{aligned} \boldsymbol{s}_t^{t+1}(t) &= H\Big(\boldsymbol{W}^t \boldsymbol{E}_3 \boldsymbol{s}_t^t(t) + \boldsymbol{R}^t \boldsymbol{V}^{t-1} \boldsymbol{s}_{t-1}^{t+1}(t-1) + \boldsymbol{b}_{\boldsymbol{W}}^t\Big) \\ &\overset{\text{ind.}}{=} H\Big(\boldsymbol{W}^t \boldsymbol{E}_3 \boldsymbol{z}_t + \boldsymbol{R}^t \boldsymbol{V}^{t-1} \boldsymbol{g}_{t-1} + \boldsymbol{b}_{\boldsymbol{W}}^t\Big) \end{aligned}$$

2. We again proceed by induction starting at $t = 1$ with (11). Observe that

$$\begin{aligned}\boldsymbol{V}^t\boldsymbol{g}_t &= \boldsymbol{V}^t H\left(\boldsymbol{U}^t H\Big(\boldsymbol{W}^t\boldsymbol{E}_3\boldsymbol{z}_t + \boldsymbol{R}^t\boldsymbol{V}^{t-1}\boldsymbol{g}_{t-1} + \boldsymbol{b}_{\boldsymbol{W}}^t\Big) + \boldsymbol{b}_U^t\right)\\ &= \Phi^{(t)}\big(\boldsymbol{E}_3\boldsymbol{z}_t, \boldsymbol{V}^{t-1}\boldsymbol{g}_{t-1}\big),\end{aligned}$$

where the inputs to $\Phi^{(t)}$ satisfy

$$\underbrace{\|\boldsymbol{E}_3\boldsymbol{z}_t - \boldsymbol{x}(t)\|}_{<\delta \text{ by (13)}} + \underbrace{\left\|\boldsymbol{V}^{t-1}\boldsymbol{g}_{t-1} - \boldsymbol{y}(t-1)\right\|}_{<\epsilon_{t-1} \text{ by induction}} < 2\epsilon_{t-1}.$$

Therefore, by construction of $\Phi^{(t)}$, in particular the bound (9), it follows that

$$\left\|\boldsymbol{V}^t\boldsymbol{g}_t - \boldsymbol{y}(t)\right\| = \left\|\Phi^{(t)}\big(\boldsymbol{E}_3\boldsymbol{z}_t, \boldsymbol{V}^{t-1}\boldsymbol{g}_{t-1}\big) - f_t\big(\boldsymbol{x}(t), \boldsymbol{y}(t-1)\big)\right\| < \epsilon_t.$$

□

- **Step 5: Define extraction layers.** We define the extraction layers similarly as in Step 5 of the proof of Theorem D.7 (see Appendix D.2), with the only difference that we do not take the sum over all blocks. This can be done with 3 layers, so that the activations of layer $T + 5$ satisfy

$$\boldsymbol{s}^{T+5}(t) = \begin{bmatrix}\mathbf{0}\\ \vdots\\ \mathbf{0}\\ \boldsymbol{g}_t\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{bmatrix}.$$

Taking the last layer weight matrix to be

$$\boldsymbol{A}^{T+6} := \begin{bmatrix}\boldsymbol{V}^1 & \dots & \boldsymbol{V}^T\end{bmatrix}, \boldsymbol{P}^{T+6} := \mathbf{0}, \boldsymbol{b}^{T+6} := \mathbf{0}$$

gives the final output

$$\boldsymbol{z}(t) = \boldsymbol{s}^{T+6}(t) = \boldsymbol{A}^{T+6}\boldsymbol{s}^{T+5}(t) = \boldsymbol{V}^t\boldsymbol{g}_t,$$

which satisfies the desired approximation, namely

$$\sum_{t=1}^{T}\|\boldsymbol{z}(t) - \boldsymbol{y}(t)\| = \sum_{t=1}^{T}\left\|\boldsymbol{V}^t\boldsymbol{g}_t - \boldsymbol{y}(t)\right\| < \sum_{t=1}^{T}\epsilon_t$$

provided that ϵ_t's are chosen sufficiently small in relative to ϵ.

□

E Theoretical and Additional experimental Evidence for Advantages of Recurrent Connections

This section complements Section E.1 of the main paper by further supporting the claim that recurrent connections enhance the representational power of (R)SNNs. In particular, Subsection E.1 presents a theoretical result on the benefits of recurrent connections, followed by the corresponding proofs in Subsection E.2 and E.3. Finally, Subsection E.4 shows experimental results which further support the claim in Section 4.

E.1 A THEORETICAL PERSPECTIVE ON THE BENEFITS OF RECURRENT CONNECTIONS

In this section, we investigate the benefits of the recurrent connections in RSNNs, i.e. the recurrent terms $\boldsymbol{R}^\ell \boldsymbol{s}^\ell(t-1)$ appearing in the dynamics (1). For simplicity, we focus on a specific setting that is popular in practical SNN implementations Eshraghian et al. (2023); Fang et al. (2021), described by:

(H1) The leakage parameter $\beta = 1$ (i.e. we have non-leaky IF neurons);

(H2) The input current is constant over time, i.e. $\boldsymbol{x}(t) = \boldsymbol{x} \in \{0,1\}^{n_0}$ for all $t \in [T]$;

(H3) $\boldsymbol{u}^\ell(0) = 0$, $\boldsymbol{b}^{(l)} = 0$ for all $l \in [L]$.

In particular, we consider the following neuron dynamics

$$\begin{cases} \boldsymbol{i}^\ell(t) = \boldsymbol{W}^\ell \boldsymbol{s}^{\ell-1}(t) + \boldsymbol{R}^\ell \boldsymbol{s}^\ell(t-1) \\ \boldsymbol{s}^\ell(t) = H\big(\boldsymbol{u}^\ell(t-1) + \boldsymbol{i}^\ell(t) - \vartheta^\ell \mathbf{1}_{n_\ell}\big) \\ \boldsymbol{u}^\ell(t) = \Big(\boldsymbol{u}^\ell(t-1) + \boldsymbol{i}^\ell(t) - \vartheta^\ell\Big) \odot \big(1 - \boldsymbol{s}^\ell(t)\big), \end{cases} \tag{14}$$

where $\boldsymbol{R}^\ell = \mathbf{0}$ refers to the case of *plain* SNNs (i.e. RSNNs but without recurrent connections). We show that in this setting, shallow plain SNNs (i.e. SNNs with no recurrent connections) cannot realize any sequence from a relatively large subset of the output spike patterns, given by

$$S_T^{\text{prime}} := \Big\{ \big(y(t)\big)_{t\in[T]} \in \{0,1\}^T : \ y(1) = 1, y(p-1) = 1 \text{ and } y(p) = 0 \text{ for all } p \in \mathbb{P} \setminus \{2\} \Big\},$$

while shallow RSNNs can realize at least a large subset of S_T^{prime}. Here $\mathbb{P} \subset \mathbb{N}$ denotes the set of prime numbers and $p_n \in \mathbb{P}$ denotes the n-th prime number, $n \in \mathbb{N}$.

Theorem E.1 (Plain SNNs vs. RSNNs)**.** *Let $\vartheta \geq 0$, $n_0, n_1, T \in \mathbb{N}$ with $T > p_{n_1+2}$ (where p_n denotes the n-th prime number), and assume (H1), (H2) and (H3) hold. Then we have*

(1) There exists no shallow plain IF-SNN with input dimension n_0, output dimension 1 and n_1 hidden neurons that can output any sequence in S_T^{prime}.

(2) Let $n_1 \geq 3$. There exists a shallow IF-RSNN with input dimension n_0, output dimension 1 and n_1 hidden neurons that can output sequences from a subset of S_T^{prime} with cardinality of at least $\binom{\lfloor \frac{T-3}{6} \rfloor}{n_1-2}$.

This result reveals an intrinsic limitation in the expressivity of shallow plain SNNs over long time horizons. In fact, they cannot realize any sequence from a relatively large subset S_T^{prime}, which has cardinality of at least 2^{T-2n_1+2}. On the other hand, as the proof reveals, shallow RSNNs can, from only one fixed single input sequence, generate a subset of S_T^{prime} with a cardinality of at least $\binom{\lfloor \frac{T-3}{6} \rfloor}{n_1-2}$. In case n_1 is large, we can further simplify this number of generated output sequences as

$$\binom{\lfloor \frac{T-3}{6} \rfloor}{n_1 - 2} \sim \frac{T^{n_1}}{n_1!} \sim \frac{T^{n_1}}{\sqrt{n_1}(n_1/e)^{n_1}} \sim \frac{(Te)^{n_1}}{\sqrt{n_1} n_1^{n_1}}.$$

Due to the prime scaling $T \geq p_{n_1+2} \sim n_1 \log n_1$, we can further estimate

$$\log \binom{\lfloor \frac{T-3}{6} \rfloor}{n_1 - 2} \sim n_1 \log(T) - n_1 \log n_1 \gtrsim n_1 \log n_1.$$

E.2 PROOF OF THEOREM E.1.1

In this subsection, we prove the first part of Theorem E.1. We starts with an auxiliary result which analyzes the periodicity of the output of a single IF neuron under the hypothesis (H2) that the inputs are constant sequences, i.e. $\boldsymbol{x}(t) = \boldsymbol{x} \in \{0,1\}^{n_0}$.

Lemma E.2 (Periodicity of a single IF neuron)**.** *Let $\boldsymbol{x} \in \{0,1\}^{n_0}, \vartheta \in \mathbb{R}_+$ and $\boldsymbol{w} \in \mathbb{R}^{n_0}$. Let us consider a single IF spiking neuron described by*

$$\begin{cases} u(0) = 0 \\ i(t) = \boldsymbol{w}^\top \boldsymbol{x} \\ s(t) = H\big(u(t-1) + i(t) - \vartheta\big) \\ u(t) = (u(t-1) + i(t)) \odot s(t), \end{cases} \tag{15}$$

for $t \in [T]$ (where $s(t)$ is the output spike sequence). Then, it holds:

1. *If $\boldsymbol{w}^\top \boldsymbol{x} \leq 0$, the output spiking sequence satisfies $s(t) = 0$ for all $t \geq 0$.*
2. *If $\boldsymbol{w}^\top \boldsymbol{x} > 0$, the output spiking sequence $s(t)$ is $\lceil \frac{\vartheta}{\boldsymbol{w}^\top \boldsymbol{x}} \rceil$-periodic. More specifically, it holds*

$$s(t) = \begin{cases} 1 & \text{if } t = k \lceil \frac{\vartheta}{\boldsymbol{w}^\top \boldsymbol{x}} \rceil \quad \text{for } k \in \mathbb{N} \\ 0 & \text{otherwise.} \end{cases}.$$

Proof. We define the time to the first spike $T_1 := T_1(\boldsymbol{x}, \boldsymbol{w}, \vartheta) := \min\{t \geq 1 : t\, \boldsymbol{w}^\top \boldsymbol{x} \geq \vartheta\}$, with the convention that $T_1 = \infty$ if the neuron never spikes. We divide the proof into two cases:

1. **Case $\boldsymbol{w}^\top \boldsymbol{x} \leq 0$:** Since $\vartheta > 0$ and $u(0) = 0$, it is trivial to see that $T_1 = \infty$. Thus, $s(t) = 0$ for all $t \geq \in [T]$.
2. **Case $\boldsymbol{w}^\top \boldsymbol{x} > 0$:** Observe that for every $t < \lceil \frac{\vartheta}{\boldsymbol{w}^\top \boldsymbol{x}} \rceil$, we have $s(t) = 0$ and $u(t) = t\boldsymbol{w}^\top \boldsymbol{x}$. At $t = \lceil \frac{\vartheta}{\boldsymbol{w}^\top \boldsymbol{x}} \rceil$, it holds

$$u(t-1) + i(t) = t\boldsymbol{w}^\top \boldsymbol{x} = \left\lceil \frac{\vartheta}{\boldsymbol{w}^\top \boldsymbol{x}} \right\rceil \boldsymbol{w}^\top \boldsymbol{x} > \vartheta.$$

   Therefore, $s\left(\lceil \frac{\vartheta}{w^\top x} \rceil\right) = 1$ and $T_1 = \lceil \frac{\vartheta}{w^\top x} \rceil$. Due to the reset-to-zero mechanism, it holds $u\big(T_1\big) = 0$. Thus, one could repeat the above argument for $t + kT_1$ with $t < T_1$ and $k \in \mathbb{N}$ to obtain

$$s(t) = \begin{cases} 1 & \text{if } t = kT_1 \quad \text{for } k \in \mathbb{N}, \\ 0 & \text{otherwise.} \end{cases}$$

□

Now we are ready to prove the first part of Theorem E.1, the statement of which, for convenience, is repeated as follows.

Theorem E.3. *Let $\vartheta \geq 0$, $n_0, n_1, T \in \mathbb{N}$ with $T > p_{n_1+2}$ (where p_n denotes the n-th prime number), and assume (H1), (H2) and (H3) hold. Then there exists no shallow plain IF-SNN with input dimension n_0, output dimension 1 and n_1 hidden neurons that can output any sequence in S_T^{prime}.*

Proof. Toward a contradiction, assume that there exist an input $\boldsymbol{x} \in \{0,1\}^{n_0}$ and a plain SNN given by $\boldsymbol{W}^1 \in \mathbb{R}^{n_1 \times n_0}$ and $\boldsymbol{w}^2 \in \mathbb{R}^{n_1}$ such that $\big(s^2(t)\big)_{t\in[T]} \in S_T^{\text{prime}}$. For each neuron $j \in [n_1]$ of the hidden layer, by Lemma E.2 there are only 2 cases, either $s_j^1(t) = 0$ for all $t \in [T]$, or $(s_j^1(t))_t$ is periodic with some period $T_j \in \mathbb{N}$ and $s_j^1(T_j) = 1$.

For any neurons j with $s_j^1(t) = 0$ for all t, then those j-th neurons do not contribute to the computation of u^2 and s^2, so we can simply ignore them (or in other words, we can assume without loss of generality that they do not exist).

Now consider the other neurons, indexed by $J \subseteq [n_1]$, which correspond to a T_j-periodic spike sequence $(s_j^1(t))_t$. Define

$$J_1 := \left\{ j \in J : (s_j^1(t))_t \text{ is 1-periodic} \right\},$$
$$J_1^C := J \setminus J_1.$$

Note that for any $j \in J_1^C$, by Lemma E.2 it holds

$$s_j^1(t) = \begin{cases} 1 & \text{if } t = kT_j,\ k \in \mathbb{N}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Due to $s^2(1) = 1$ and the 1-periodicity of $\big(s_j^1(t)\big)_{t\in[T]}$, it holds for every $t \in [T]$ that

$$\vartheta \leq \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1(1)\right\rangle = \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1_{J_1}(1)\right\rangle + \underbrace{\left\langle \boldsymbol{w}^2, \boldsymbol{s}^1_{J_1^C}(1)\right\rangle}_{=0 \text{ by (16)}} = \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1_{J_1}(t)\right\rangle.$$

On the other hand, for $j \in J_1^C$ and $t \in \mathbb{P}$, unless $t = T_j$ holds and T_j is a prime number, t cannot be decomposed into $t = kT_j$ and hence $s_j^1(t)$ must be 0. By the assumption $T > p_{n_1+2}$, there exists a prime $p \in \big(\mathbb{P} \setminus \{2\}\big) \cap [T]$ (note that this set has a cardinality of at least $n_1 + 1$) such that $p \notin \big\{T_j : j \in J_1^C\big\}$ (since the latter set has its cardinality bounded by n_1). For this particular choice of p, we have $s_j^1(p) = 0$ for every $j \in J_1^C$. Due to $s^2(p-1) = 1$ we know that $u^2(p-1) = 0$, which combined with $s^2(p) = 0$ and the choice of p yields

$$\vartheta > \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1(p)\right\rangle = \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1_{J_1}(p)\right\rangle + \underbrace{\left\langle \boldsymbol{w}^2, \boldsymbol{s}^1_{J_1^C}(p)\right\rangle}_{=0} = \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1_{J_1}(p)\right\rangle,$$

which contradicts the previous equation. □

E.3 Proof of Theorem E.1.2

Now we prove the second part of Theorem E.1, which is re-stated as follows.

Theorem E.4. *Let $\vartheta \geq 0$, $n_0, n_1, T \in \mathbb{N}$ with $T > p_{n_1+2}$, $n_1 \geq 3$, and assume (H1), (H2) and (H3) hold. Let $n_1 \geq 3$. There exists a shallow IF-RSNN with input dimension n_0, output dimension 1 and n_1 hidden neurons that can output sequences from a subset of S_T^{prime} with cardinality of at least $\binom{\lfloor \frac{T-3}{6} \rfloor}{n_1-2}$.*

Proof. Without loss of generality, we assume that $n_0 = 1$, as for $n_0 > 1$ we can zero out the connections to the i-th input neurons, $i \geq 2$. We fix the input sequence to be $x(t) := x := 1$, $t \in [T]$.

Input to hidden layer: First, we define $w_1^1 := \vartheta$, $R_{11}^1 := -T\vartheta$ and $R_{1,j}^1 := 0$ for $j \in [n_1] \setminus \{1\}$, so that

$$s_1^1(t) = \begin{cases} 1, & t = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Second, setting $w_2^1 := \vartheta/2$, $R_{2,j}^1 := 0$ for all $j \in [n_1]$, we obtain

$$s_2^1(t) = \begin{cases} 1, & t = 2k, \\ 0, & \text{otherwise.} \end{cases}$$

Now we turn to the remaining $(n_1 - 2)$ neurons that are left in the hidden layer. For any $j \in \{2, \ldots, n_1\}$, we aim to obtain

$$s_j^1(t) = \begin{cases} 1, & t = 6k_j + 3, \\ 0, & \text{otherwise,} \end{cases}$$

where $k_j \in \left[\lfloor \frac{T-3}{6} \rfloor \right]$. For this, we set $w_j^1 := \frac{\vartheta}{6k_j+3}$, $R_{j,j}^1 := -T\vartheta$ and $R_{j,j'}^1 := 0$ for $j' \in [n_1]\setminus\{j\}$.

Note that there are $\binom{\lfloor \frac{T-3}{6} \rfloor}{n_1-2}$ ways to choose a subset $\{k_j : j \in \{2, \ldots, n_1 - 2\}\}$ from the set $\left[\lfloor \frac{T-3}{6} \rfloor \right]$ (order does not matter).

Hidden layer to output layer: We realize an time-step-wise OR layer. For this, we set $\boldsymbol{w}^2 := \vartheta \mathbf{1}_{n_1}$ and $R^2 := 0$. We obtain

$$s^2(t) = H\Big(u^2(t-1) + \left\langle \boldsymbol{w}^2, \boldsymbol{s}^1(t)\right\rangle - \vartheta\Big) = 1.$$

Then $u^2(t) = 0$ holds by induction for every $t \in [T]$. First, observe that it holds for $t = 1$ because $s_1^1(1) = 1$ and hence $u^2(1) = 0$ be the reset-to-zero rule. Furthermore, at time step t, assume that

$u^2(t-1) = 0$, we consider two cases. If $s^2(t) = 1$, then again the membrane potential is reset to 0. Otherwise, $s^2(t) = 0$, it means that $\langle \boldsymbol{w}^2, \boldsymbol{s}^1(t) \rangle < \vartheta$, and hence $\boldsymbol{s}^1(t) = \mathbf{0}_{n_1}$. In other words, unless $\boldsymbol{s}^1(t) = \mathbf{0}_{n_1}$, which yields $s^2(t) = 0$, we get $s^2(t) = 1$.

Combining this construction with the previous construction of $\big(\boldsymbol{s}^1(t)\big)_{t\in[T]}$, we get that

$$s^2(t) = \begin{cases} 1, & t \in \{1, 2k, 6k_j + 3\}_{k,k_j\in\mathbb{N}} \cap [T], \\ 0, & \text{else.} \end{cases}$$

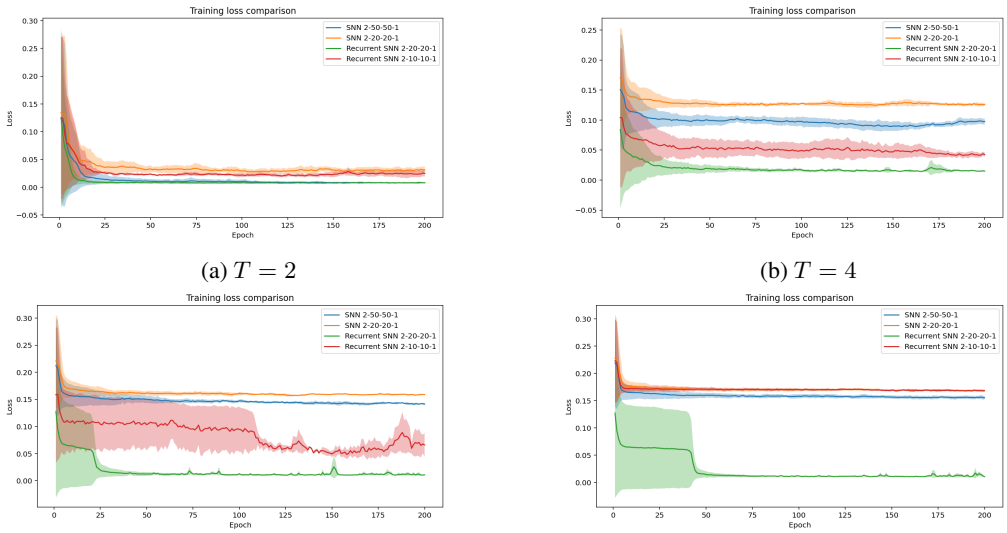
In particular, it holds $s^2(1) = 1$. Also, for any $p \in \mathbb{P} \setminus \{2\}$, we have $s^2(p-1) = 1$ (because $p-1$ is odd) and $s^2(p) = 0$ (because p is not divisible by 2 or 3, while $2k$ is divisible by 2 and $6k_j + 3$ is divisible by 3). Thus $\big(s^2(t)\big)_{t\in[T]}$ belongs to S_T^{prime}.

On the other hand, since there are $\binom{\lfloor \frac{T-3}{6} \rfloor}{n_1-2}$ ways to choose $\{k_j : j \in \{2, \ldots, n_1\}\}$, each of which leads to a different output sequence $\big(s^2(t)\big)_{t\in[T]} \in S_T^{\text{prime}}$. This shows that RSNNs can realize at least this number of output sequences from S_T^{prime} just from a single constant input sequence.

□

E.4 Additional experimental results

We vary the experimental setup described in Section 4 to increase the reliability and diversity of the empirical observations. In particular, we modify the target output sequences $\big(y(t)\big)t \in [T]$ by setting $y^{(i)}(t) = x^{(i)}_{p(t)}(t)$, where $p(t)$ denotes the coordinate indicator defined previously.



(a) $T = 2$ (b) $T = 4$

(c) $T = 8$ (d) $T = 16$

Figure 2: The learning curves of SNNs and RSNNs of various sizes, trained on a simple synthetic dataset and averaged over 5 runs. The plain SNNs have 501 and 2751 trainable parameters, the RSNNs have 2101 and 551 trainable parameters.

Figure 2 shows the comparison of training losses achieved by two plain SNNs (each with 2 hidden layers of width $n \in \{20, 50\}$) and two recurrent RSNNs (each also with 2 hidden with $n' \in \{10, 20\}$). In general, the smaller plain SNN (orange) leads to the highest training loss, while incorporating recurrent connections to the same spatial architecture (green) yields the lowest loss.

By increasing the width of the hidden layers in the plain SNN to $n = 50$ (blue), one could reduce the loss but at the expense of significantly enlarging the number of trainable parameters (from 501 to 2751). Meanwhile, one could also add recurrent connections and reduce the hidden layer width to

compensate for the increase in the number of parameters (red). This generally fits the training data better than the original plain SNN, except for $T = 16$, where T is even larger than the hidden layer width.