Journal of Machine Learning Research 24 (2023) 1-41 Submitted 10/22; Published 2/23

Minimal Width for Universal Property of Deep RNN

Chang hoon Song GOLDBACH2@SNU.AC.KR
Department of Mathematical Science
Seoul National University

Geonho Hwang HGH2134@SNU.AC.KR
Department of Mathematical Science
Seoul National University

Jun ho Lee JHLSAT@KONGJU.AC.KR
Kongju National Universality

Myungjoo Kang MKANG@SNU.AC.KR
Department of Mathematical Science
Seoul National University

Editor: Mehryar Mohri

Abstract

A recurrent neural network (RNN) is a widely used deep-learning network for dealing with sequential data. Imitating a dynamical system, an infinite-width RNN can approximate any open dynamical system in a compact domain. In general, deep narrow networks with bounded width and arbitrary depth are more effective than wide shallow networks with arbitrary width and bounded depth in practice; however, the universal approximation theorem for deep narrow structures has yet to be extensively studied. In this study, we prove the universality of deep narrow RNNs and show that the upper bound of the minimum width for universality can be independent of the length of the data. Specifically, we show a deep RNN with ReLU activation can approximate any continuous function or L^p function with the widths $d_x + d_y + 3$ and $\max\{d_x + 1, d_y\}$, respectively, where the target function maps a finite sequence of vectors in $\mathbb{R}^{d_x}$ to a finite sequence of vectors in $\mathbb{R}^{d_y}$. We also compute the additional width required if the activation function is sigmoid or more. In addition, we prove the universality of other recurrent networks, such as bidirectional RNNs. Bridging a multi-layer perceptron and an RNN, our theory and technique can shed light on further research on deep RNNs.

Keywords: recurrent neural network, universal approximation, deep narrow network, sequential data, minimum width

1. Introduction

Deep learning, a type of machine learning that approximates a target function using a numerical model called an artificial neural network, has shown tremendous success in diverse fields, such as regression (Garnelo et al., 2018), image processing (Dai et al., 2021; Zhai et al., 2022), speech recognition (Baevski et al., 2020), and natural language processing (LeCun et al., 2015; Lavanya and Sasikala, 2021).

While the excellent performance of deep learning is attributed to a combination of various factors, it is reasonable to speculate that its notable success is partially based on

the *universal approximation theorem*, which states that neural networks are capable of arbitrarily accurate approximations of the target function.

Formally, for any given function f in a target class and $\epsilon > 0$, there exists a network $\mathcal{N}$ such that $\|f - \mathcal{N}\| < \epsilon$. In topological language, the theorem states that a set of networks is dense in the class of the target function. In this sense, the closure of the network defines the range of functions it network can represent.

As universality provides the expressive power of a network structure, studies on the universal approximation theorem have attracted increasing attention. Examples include the universality of multi-layer perceptrons (MLPs), the most basic neural networks (Cybenko, 1989; Hornik et al., 1989; Leshno et al., 1993), and the universality of recurrent neural networks (RNNs) with the target class of open dynamical systems (Schäfer and Zimmermann, 2007). Recently, Zhou (2020) has demonstrated the universality of convolutional neural networks.

Classical universal approximation theorems specialize in the representation power of shallow wide networks with bounded depth and unbounded width. Based on mounting empirical evidence that deep networks demonstrate better performance than wide networks, the construction of deep networks instead of shallow networks has gained considerable attention in recent literature. Consequently, researchers have started to analyze the universal approximation property of deep networks (Cohen et al., 2016; Lin and Jegelka, 2018; Rolnick and Tegmark, 2018; Kidger and Lyons, 2020). Studies on MLP have shown that wide shallow networks require only two depths to have universality, while deep narrow networks require widths at least as their input dimension.

A wide network obtains universality by increasing its width even if the depth is only two (Cybenko, 1989; Hornik et al., 1989; Leshno et al., 1993). However, in the case of a deep network, there is a function for a narrow network that cannot be able to approximated, regardless of its depth (Lu et al., 2017; Park et al., 2021). Therefore, clarifying the *minimum width* to guarantee universality is crucial, and studies are underway to investigate its lower and upper bounds, narrowing the gap.

Recurrent neural networks (RNNs) (Rumelhart et al., 1986; Elman, 1990) have been crucial for modeling complex temporal dependencies in sequential data. They have various applications in diverse fields, such as language modeling (Mikolov et al., 2010; Jozefowicz et al., 2016), speech recognition (Graves and Jaitly, 2014; Bahdanau et al., 2016), recommendation systems (Hidasi et al., 2015; Wu et al., 2017), and machine translation (Bahdanau et al., 2014). Deep RNNs are widely used and have been successfully applied in practical applications. However, their theoretical understanding remains elusive despite their intensive use. This deficiency in existing studies motivated our work.

In this study, we prove the universal approximation theorem of deep narrow RNNs and discover the upper bound of their minimum width. The target class consists of a sequence-to-sequence function that depends solely on past information. We refer to such functions as *past-dependent* functions. We provide the upper bound of the minimum width of the RNN for universality in the space of the past-dependent functions. Surprisingly, the upper bound is independent of the length of the sequence. This theoretical result highlights the suitability of the recurrent structure for sequential data compared with other network structures. Furthermore, our results are not restricted to RNNs; they can be generalized to variants of RNNs, including *long short-term memory* (LSTM), *gated recurrent units* (GRU),

Network	Function class	Activation	Result
RNN	$C\left(\mathcal{K}, \mathbb{R}^{d_y}\right)^\dagger$	ReLU	$w_{\min} \leq d_x + d_y + 3$
		conti. nonpoly[1]	$w_{\min} \leq d_x + d_y + 3$
		conti. nonpoly[2]	$w_{\min} \leq d_x + d_y + 4$
	$L^p\left(\mathcal{K}, \mathbb{R}^{d_y}\right)^\dagger$	conti. nonpoly[1,2]	$w_{\min} \leq \max\{d_x + 1, d_y\} + 1$
	$L^p\left(\mathbb{R}^{d_x}, \mathbb{R}^{d_y}\right)^\dagger$	ReLU	$w_{\min} = \max\{d_x + 1, d_y\}$
LSTM	$C\left(\mathcal{K}, \mathbb{R}^{d_y}\right)^\dagger$		$w_{\min} \leq d_x + d_y + 3$
	$L^p\left(\mathcal{K}, \mathbb{R}^{d_y}\right)^\dagger$		$w_{\min} \leq \max\{d_x + 1, d_y\} + 1$
GRU	$C\left(\mathcal{K}, \mathbb{R}^{d_y}\right)^\dagger$		$w_{\min} \leq d_x + d_y + 3$
	$L^p\left(\mathcal{K}, \mathbb{R}^{d_y}\right)^\dagger$		$w_{\min} \leq \max\{d_x + 1, d_y\} + 1$
BRNN	$C\left(\mathcal{K}, \mathbb{R}^{d_y}\right)$	ReLU	$w_{\min} \leq d_x + d_y + 3$
		conti. nonpoly[1]	$w_{\min} \leq d_x + d_y + 3$
		conti. nonpoly[2]	$w_{\min} \leq d_x + d_y + 4$
	$L^p\left(\mathcal{K}, \mathbb{R}^{d_y}\right)$	conti. nonpoly[1,2]	$w_{\min} \leq \max\{d_x + 1, d_y\} + 1$
	$L^p\left(\mathbb{R}^{d_x}, \mathbb{R}^{d_y}\right)$	ReLU	$w_{\min} \leq \max\{d_x + 1, d_y\}$

† requires the class to consists of past-dependent functions.
[1] requires an activation σ to be continuously differentiable at some point z_0 with $\sigma(z_0) = 0$ and $\sigma'(z_0) \neq 0$. tanh belongs here.
[2] requires an activation σ to be continuously differentiable at some point z_0 with $\sigma'(z_0) \neq 0$. A logistic sigmoid function belongs here.

Table 1: Summary of our results on the upper bound of the minimal width $w_{\min}$ for the universality of RNNs. In the table, $\mathcal{K}$ indicates a compact subset of $\mathbb{R}^{d_x}$ and $1 \leq p < \infty$. We abbreviate continuous to "conti".

and *bidirectional RNNs* (BRNN). As corollaries of our main theorem, LSTM and GRU are shown to have the same universality and target class as an RNN. We also prove that the BRNN can approximate any sequence-to-sequence function in a continuous or L^p space under the respective norms. We also present the upper bound of the minimum width for these variants. Table 1 outlines our main results.

1.1 Summary of Contributions

With a target class of functions that map a finite sequence $x \in \mathbb{R}^{d_x \times N}$ to a finite sequence $y \in \mathbb{R}^{d_y \times N}$, we prove the following:

- A deep RNN can approximate any past-dependent sequence-to-sequence continuous function with width $d_x + d_y + 3$ for the ReLU or tanh[1], and $d_x + d_y + 4$ for non-degenerating activations.

- A deep RNN can approximate any past-dependent L^p function ($1 \leq p < \infty$) with width $\max\{d_x + 1, d_y\}$ for the ReLU activation and $\max\{d_x + 1, d_y\} + 1$ for non-degenerating activations.

- A deep BRNN can approximate any sequence-to-sequence continuous function with width $d_x + d_y + 3$ for the ReLU or tanh[1], and $d_x + d_y + 4$ for non-degenerating activations.

- A deep BRNN can approximate any sequence-to-sequence L^p function ($1 \leq p < \infty$) with width $\max\{d_x + 1, d_y\}$ for the ReLU activation and $\max\{d_x + 1, d_y\} + 1$ for non-degenerating activations.

- A deep LSTM or GRU can approximate any past-dependent sequence-to-sequence continuous function with width d_x+d_y+3 and L^p function with width $\max\{d_x + 1, d_y\}+1$.

1.2 Organization

In Section 2, we briefly review previous studies on the universality of neural networks. Section 3 provides some notations, network configuration, a description of the target function class, and the condition of the activation function. Section 4 addresses the approximation property in a continuous function space. First, we fit only the last component of an output sequence and extend it to all components of the sequence. In Section 5, we present the upper bound of the minimum width in L^p space. As an expansion of our theorems, the universal approximation theorems for the LSTM, GRU, and BRNN are deduced in Section 6. Sections 7 and 8 present a discussion and conclusions, respectively.

2. Related Works

We briefly review some of the results of studies on the universal approximation property. Studies have been conducted to determine whether a neural network can learn a sufficiently wide range of functions, that is, whether it has universal properties. Cybenko (1989) and Hornik et al. (1989) first proved that the most basic network, a simple two-layered MLP, can approximate arbitrary continuous functions defined on a compact set. Some follow-up studies have investigated the universal properties of other structures for a specific task, such as a convolutional neural network for image processing (Zhou, 2020), an RNN for open dynamical systems (Schäfer and Zimmermann, 2007; Hanson and Raginsky, 2020), and transformer networks for translation and speech recognition (Yun et al., 2020). Particularly for RNNs, Schäfer and Zimmermann (2007) showed that an open dynamical system with continuous state transition and output function can be approximated by a network with a wide RNN cell and subsequent linear layer in finite time. Hanson and Raginsky (2020) showed that the trajectory of the dynamical system can be reproduced with arbitrarily small errors up to infinite time, assuming a stability condition on long-term behavior.

While such prior studies mainly focused on wide and shallow networks, several studies have determined whether a deep narrow network with bounded width can approximate arbitrary functions in some kinds of target function classes, such as a space of continuous functions or an L^p space (Lu et al., 2017; Hanin and Sellke, 2017; Johnson, 2019; Kidger and Lyons, 2020; Park et al., 2021). Unlike the case of a wide network that requires only

1. Generally, non-degenerate σ with $\sigma(z_0) = 0$ requires the same minimal width as tanh.

one hidden layer for universality, there exists a function that cannot be approximated by any network whose width is less than a certain threshold. Specifically, in a continuous function space $C\left(K, \mathbb{R}^{d_y}\right)$ with the supreme-norm $\|f\|_\infty := \sup_{x\in K} \|f(x)\|$ on a compact subset $K \subset \mathbb{R}^{d_x}$, Hanin and Sellke (2017) showed negative and positive results that MLPs do not attain universality with width d_x. Still, width $d_x + d_y$ is sufficient for MLPs to achieve universality with the ReLU activation function. Johnson (2019); Kidger and Lyons (2020) generalized the condition on the activation and proved that d_x is too narrow, but $d_x + d_y + 2$ is sufficiently wide for the universality in $C\left(K, \mathbb{R}^{d_y}\right)$. On the other hand, in an L^p space $L^p\left(\mathbb{R}^{d_x}, \mathbb{R}^{d_y}\right)$ with L^p-norm $\|f\|_p := \left(\int_{\mathbb{R}^{d_x}} f(x)\, dx\right)^{1/p}$ for $p \in [0, \infty)$, Lu et al. (2017) showed that the width d_x is insufficient for MLPs to have universality, but $d_x + 4$ is enough, with the ReLU activation when $p = 1$ and $d_y = 1$. Kidger and Lyons (2020) found that MLPs whose width is $d_x + d_y + 1$ achieve universality in $L^p\left(\mathbb{R}^{d_x}, \mathbb{R}^{d_y}\right)$ with the ReLU activation, and Park et al. (2021) determined the exact value $\max\{d_x + 1, d_y\}$ required for universality in the L^p space with ReLU.

As described earlier, studies on deep narrow MLP have been actively conducted, but the approximation ability of deep narrow RNNs remains unclear. This is because the process by which the input affects the result is complicated compared with that of an MLP. The RNN cell transfers information to both the next time step in the same layer and the same time step in the next layer, which makes it difficult to investigate the minimal width. In this regard, we examined the structure of the RNN to apply the methodology and results from the study of MLPs to deep narrow RNNs.

3. Terminologies and Notations

This section introduces the definition of network architecture and the notation used throughout this paper. d_x and d_y denote the dimension of input and output space, respectively. σ is an activation function unless otherwise stated. Sometimes, v indicates a vector with suitable dimensions.

First, we used square brackets, subscripts, and colon symbols to index a sequence of vectors. More precisely, for a given sequence of d_x-dimensional vectors $x : \mathbb{N} \to \mathbb{R}^{d_x}$, $x[t]_j$ or $x_j[t]$ denotes the j-th component of the t-th vector. The colon symbol : is used to denote a continuous index, such as $x[a : b] = (x[i])_{a\le i\le b}$ or $x[t]_{a:b} = (x[t]_a, x[t]_{a+1}, \ldots, x[t]_b)^T \in \mathbb{R}^{b-a+1}$. We call the sequential index t by time and each $x[t]$ a *token*.

Second, we define the token-wise linear maps $\mathcal{P} : \mathbb{R}^{d_x\times N} \to \mathbb{R}^{d_s\times N}$ and $\mathcal{Q} : \mathbb{R}^{d_s\times N} \to \mathbb{R}^{d_y\times N}$ to connect the input, hidden state, and output space. As the dimension of the hidden state space $\mathbb{R}^{d_s}$ on which the RNN cells act is different from those of the input domain $\mathbb{R}^{d_x}$ and output domain $\mathbb{R}^{d_y}$, we need maps adjusting the dimensions of the spaces. For a given matrix $P \in \mathbb{R}^{d_s\times d_x}$, a *lifting map* $\mathcal{P}(x)[t] := Px[t]$ lifts the input vector to the hidden state space. Similarly, for a given matrix $Q \in \mathbb{R}^{d_y\times d_s}$, a *projection map* $\mathcal{Q}(s)[t] := Qs[t]$ projects a hidden state onto the output vector. As the first token defines a token-wise map, we sometimes represent token-wise maps without a time length, such as $\mathcal{P} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_s}$ instead of $\mathcal{P} : \mathbb{R}^{d_x\times N} \to \mathbb{R}^{d_s\times N}$.

Subsequently, an RNN is constructed using a composition of basic recurrent cells between the lifting and projection maps. We considered four basic cells: RNN, LSTM, GRU, and BRNN.

- **RNN Cell** A *recurrent cell*, *recurrent layer*, or *RNN cell* $\mathcal{R}$ maps an input sequence $x = (x[1], x[2], \ldots) = (x[t])_{t\in\mathbb{N}} \in \mathbb{R}^{d_s\times\mathbb{N}}$ to an output sequence $y = (y[t])_{t\in\mathbb{N}} \in \mathbb{R}^{d_s\times\mathbb{N}}$ using

$$y[t+1] = \mathcal{R}(x)[t+1] = \sigma\left(A\mathcal{R}(x)[t] + Bx[t+1] + \theta\right), \tag{1}$$

where σ is an activation function, $A, B \in \mathbb{R}^{d_s\times d_s}$ are the weight matrices, and $\theta \in \mathbb{R}^{d_s}$ is the bias vector. The initial state $\mathcal{R}(x)[0]$ can be an arbitrary constant vector, which is zero vector $\mathbf{0}$ in this setting.

- **LSTM cell** *An LSTM cell* is an RNN cell variant with an internal cell state and gate structures. As an original version proposed in Hochreiter and Schmidhuber (1997), an LSTM cell includes only two gates: an input gate and an output gate. Still, the base of almost all networks used in modern deep learning is LSTM cells with an additional gate called the forget gate, introduced in Gers et al. (2000). Hence we present the LSTM cell with three gates instead of the original one.

  Mathematically, an LSTM cell $\mathcal{R}_{LSTM}$ is a process that computes an output $h \in \mathbb{R}^{d_s}$ and cell state $c \in \mathbb{R}^{d_s}$, defined by the following relation:

$$\begin{aligned}
i[t+1] &= \sigma_{\text{sig}}\left(W_i x[t+1] + U_i h[t] + b_i\right)\\
f[t+1] &= \sigma_{\text{sig}}\left(W_f x[t+1] + U_f h[t] + b_f\right)\\
g[t+1] &= \tanh\left(W_g x[t+1] + U_g h[t] + b_g\right)\\
o[t+1] &= \sigma_{\text{sig}}\left(W_o x[t+1] + U_o h[t] + b_o\right)\\
c[t+1] &= f[t+1] \odot c[t] + i[t+1] \odot g[t+1]\\
h[t+1] &= o[t+1] \odot \tanh\left(c[t+1]\right)
\end{aligned} \tag{2}$$

  where $W_* \in \mathbb{R}^{d_s\times d_s}$ and $U_* \in \mathbb{R}^{d_s\times d_s}$ are weight matrices; $b_* \in \mathbb{R}^{d_s}$ is the bias vector for each $* = i, f, g, o$; and σ_{sig} is the sigmoid activation function. $\odot$ denotes component-wise multiplication, and we set the initial state to zero in this study.

  Although some variants of the LSTM have additional connections, such as peephole connection, our result for LSTM extends naturally.

- **GRU cell** Another import variation of an RNN is GRU, proposed by Cho et al. (2014). Mathematically, a *GRU cell* $\mathcal{R}_{GRU}$ is a process that computes $h\in \mathbb{R}^{d_s}$ defined by

$$\begin{aligned}
r[t+1] &= \sigma_{\text{sig}}\left(W_r x[t+1] + U_r h[t] + b_r\right),\\
\tilde{h}[t+1] &= \tanh\left(W_h x[t+1] + U_h\left(r[t+1] \odot h[t]\right) + b_h\right),\\
z[t+1] &= \sigma_{\text{sig}}\left(W_z x[t+1] + U_z h[t] + b_z\right),\\
h[t+1] &= \left(1 - z[t+1]\right) \odot h[t] + z[t+1] \odot \tilde{h}[t+1],
\end{aligned} \tag{3}$$

  where $W_* \in \mathbb{R}^{d_s\times d_s}$ and $U_* \in \mathbb{R}^{d_s\times d_s}$ are weight matrices, $b_* \in \mathbb{R}^{d_s}$ is the bias vector for each $* = r, z, h$, and σ_{sig} is the sigmoid activation function. $\odot$ denotes component-wise multiplication, and we set the initial state to zero in this study.

- **BRNN cell** A *BRNN cell* $\mathcal{BR}$ consists of a pair of RNN cells and a token-wise linear map that follows the cells (Schuster and Paliwal, 1997). An RNN cell $\mathcal{R}_1$ in the BRNN

cell $\mathcal{BR}$ receives input from $x[1]$ to $x[N]$ and the other $\mathcal{R}_2$ receives input from $x[N]$ to $x[1]$ in reverse order. Then, the linear map $\mathcal{L}$ in $\mathcal{BR}$ combines the two outputs from the RNN cells. Specifically, a BRNN cell $\mathcal{BR}$ is defined as follows:

$$\begin{aligned} \mathcal{R}(x)[t+1] &:= \sigma\left(A\mathcal{R}_1(x)[t] + Bx[t+1] + \theta\right), \\ \bar{\mathcal{R}}(x)[t-1] &:= \sigma\left(\bar{A}\bar{\mathcal{R}}(x)[t] + \bar{B}x[t-1] + \bar{\theta}\right), \\ \mathcal{BR}(x)[t] &:= \mathcal{L}\left(\mathcal{R}(x)[t], \bar{\mathcal{R}}(x)[t]\right) \\ &:= W\mathcal{R}(x)[t] + \bar{W}\bar{\mathcal{R}}(x)[t]. \end{aligned} \tag{4}$$

where A, B, $\bar{A}$, $\bar{B}$, W, and $\bar{W}$ are weight matrices; θ and $\bar{\theta}$ are bias vectors.

- **Network architecture** An *RNN* $\mathcal{N}$ comprises a lifting map $\mathcal{P}$, projection map $\mathcal{Q}$, and L recurrent cells $\mathcal{R}_1, \ldots, \mathcal{R}_L$;

$$\mathcal{N} := \mathcal{Q} \circ \mathcal{R}_L \circ \cdots \circ \mathcal{R}_1 \circ \mathcal{P}. \tag{5}$$

  We denote the network as a *stack RNN* or *deep RNN* when $L \geq 2$, and each output of the cell $\mathcal{R}_i$ as the i-th hidden state. d_s indicates the width of the network. As a special case, a recurrent cell $\mathcal{R}$ in (1) is a composition of activation and a token-wise affine map with $A = 0$; A token-wise MLP is a specific example of RNN. If LSTM, GRU, or BRNN cells replace recurrent cells, the network is called an LSTM, a GRU, or a BRNN.

In addition to the type of cell, the activation function σ affects universality. We focus on the case of ReLU or tanh while also considering the general activation function satisfying the condition proposed by (Kidger and Lyons, 2020). σ is a continuous non-polynomial function that is continuously differentiable at some z_0 with $\sigma'(z_0) \neq 0$. We refer to the condition as a *non-degenerate* condition and z_0 as a *non-degenerating point*.

Finally, the target class must be set as a subset of the sequence-to-sequence function space, from $\mathbb{R}^{d_x}$ to $\mathbb{R}^{d_y}$. Given an RNN $\mathcal{N}$, each token $y[t]$ of the output sequence $y = \mathcal{N}(x)$ depends only on $x[1:t] := (x[1], x[2], \ldots, x[t])$ for the input sequence x. We define this property as *past dependency* and a function with this property as a *past-dependent* function. More precisely, if all the output tokens of a sequence-to-sequence function are given by $f[t](x[1:t])$ for functions $f[t] : \mathbb{R}^{d_x \times t} \to \mathbb{R}^{d_y}$, we say that the function is past-dependent. Meanwhile, we must fix the finite length or terminal time $N < \infty$ of the input and output sequence. Without additional assumptions such as in (Hanson and Raginsky, 2020), errors generally accumulate over time, making it impossible to approximate implicit dynamics up to infinite time regardless of past dependency. Therefore we set the target function class as a class of past-dependent sequence-to-sequence functions with sequence length N.

Remark 1 *On a compact domain and under bounded length, the continuity of $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ implies that of each $f[t] : \mathbb{R}^{d_x \times t} \to \mathbb{R}^{d_y}$ and vice versa. In the case of the L^p norm with $1 \leq p < \infty$, $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ is L^p integrable if and only if $f[t]$ is L^p integrable for each t. In both cases, the sequence of functions $(f_n)_{n \in \mathbb{N}}$ converges to g if and only if $(f_n[t])_{n \in \mathbb{N}}$ converges to $g[t]$ for each t. Thus, we focus on approximating $f[t]$ for each t under the given conditions.*

Remark 2 *There are typical tasks where the length of the output sequence differs from that of the input sequence. In translation tasks, for instance, inputs and outputs are sentences represented as sequences of tokens of words in different languages, and the length of each sequence differs in general. Nevertheless, even in such a case, a sequence* $x = (x[1], x[2], \dots, x[N]) \in \mathbb{R}^{d_x \times N}$ *can be naturally embedded into* $\mathbb{R}^{d_x \times M}$ *for an integer* $M > N$ *as an extended sequence* $(x[1], x[2], \dots, x[N], \mathbf{0}, \dots, \mathbf{0})$ *or* $(\mathbf{0}, \dots, \mathbf{0}, x[1], x[2], \dots, x[N])$*. By the natural embedding, a map between sequences* $f : \mathbb{R}^{d_x \times N_1} \to \mathbb{R}^{d_y \times N_2}$ *is a special case of a map between* $f : \mathbb{R}^{d_x \times M} \to \mathbb{R}^{d_y \times M}$ *for an integer* $M \geq N_1, N_2$*.*

Sometimes, only the last value $\mathcal{N}(x)[N]$ is required considering an RNN $\mathcal{N}$ as a sequence-to-vector function $\mathcal{N} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y}$. We freely use the terminology RNN for sequence-to-sequence and sequence-to-vector functions because there is no confusion when the output domain is evident.

We have described all the concepts necessary to set a problem, but we end this section with an introduction to the concepts used in the proof of the main theorem. For the convenience of the proof, we slightly modify the activation σ to act only on some components, instead of all components. With activation σ and index set $I \subseteq \mathbb{N}$, *the modified activation* σ_I is defined as

$$\sigma_I(s)_i = \begin{cases} \sigma(s_i) & \text{if } i \in I \\ s_i & \text{otherwise} \end{cases} \tag{6}$$

Using the modified activation function σ_I, the basic cells of the network are modified in (1). For example, a *modified recurrent cell* can be defined as

$$\begin{aligned} \mathcal{R}(x)[t+1]_i &= \sigma_I \left(A\mathcal{R}(x)[t] + Bx[t+1] + \theta \right)_i \\ &= \begin{cases} \sigma \left(A\mathcal{R}(x)[t] + Bx[t+1] + \theta \right)_i & \text{if } i \in I \\ \left(A\mathcal{R}(x)[t] + Bx[t+1] + \theta \right)_i & \text{otherwise} \end{cases} . \end{aligned} \tag{7}$$

Similarly, *modified RNN, LSTM, GRU, or BRNN* is defined using modified cells in (1). This concept is similar to the enhanced neuron of Kidger and Lyons (2020) in that activation can be selectively applied, but is different in that activation can be applied to partial components.

As activation leads to the non-linearity of a network, modifying the activation can affect the minimum width of the network. Fortunately, the following lemma shows that the minimum width increases by at most one owing to the modification. We briefly introduce the ideas here, with a detailed proof provided in the appendix.

Lemma 3 *Let* $\bar{\mathcal{R}} : \mathbb{R}^{d \times N} \to \mathbb{R}^{d \times N}$ *be a modified RNN cell,* $\bar{\mathcal{Q}} : \mathbb{R}^d \to \mathbb{R}^d$*, and* $\bar{\mathcal{P}} : \mathbb{R}^d \to \mathbb{R}^d$ *be a token-wise linear projection and lifting map. Suppose that an activation function* σ *of* $\bar{\mathcal{R}}$ *is non-degenerate with a non-degenerating point* z_0*. Then for any compact subset* $K \subset \mathbb{R}^d$ *and* $\epsilon > 0$*, there exists RNN cells* $\mathcal{R}_1$*,* $\mathcal{R}_2 : \mathbb{R}^{(d+\beta(\sigma)) \times N} \to \mathbb{R}^{(d+\beta(\sigma)) \times N}$*, and a token-wise linear map* $\mathcal{P} : \mathbb{R}^d \to \mathbb{R}^{d+\beta(\sigma)}$*,* $\mathcal{Q} : \mathbb{R}^{d+\beta(\sigma)} \to \mathbb{R}^d$ *such that*

$$\sup_{x \in K^N} \left\| \bar{\mathcal{Q}} \circ \bar{\mathcal{R}} \circ \bar{\mathcal{P}}(x) - \mathcal{Q} \circ \mathcal{R}_2 \circ \mathcal{R}_1 \circ \mathcal{P}(x) \right\| < \epsilon, \tag{8}$$

where $\beta(\sigma) = \begin{cases} 0 & \text{if } \sigma(z_0) = 0 \\ 1 & \text{otherwise} \end{cases}$

Proof [Sketch of proof] The detailed proof is available in Appendix B. We use the Taylor expansion of σ at z_0 to recover the value before activation. For the i-th component with $i \notin I$, choose a small $\delta > 0$ and linearly approximate $\sigma(z_0 + \delta z)$ as $\sigma(z_0) + \delta\sigma'(z_0)z$. An affine transform after the Taylor expansion recovers z. ■

Remark 4 *Since the additional width is only used to move the input of each RNN cell to near z_0, it seems easily removable using a bias term at first glance, provided the projection map is affine instead of linear. However, a bias term θ in (1) is fixed across all time steps, while we need different translations: one for the first token and the other for tokens after the second.*

For example, for an RNN cell $\mathcal{R}^\delta : \mathbb{R}^{1\times\mathbb{N}} \to \mathbb{R}^{1\times\mathbb{N}}$ defined by

$$\mathcal{R}^\delta(x)[t+1] = \sigma\left(a\mathcal{R}^\delta(x)[t] + \delta x[t+1] + \theta\right), \tag{9}$$

we need $\theta = z_0$ to use the Taylor expansion at z_0 to the first token $\mathcal{R}^\delta(x)[1] = \sigma(\delta x[1] + \theta)$. When $\theta = z_0$, we cannot represent the second token

$$\begin{aligned}\mathcal{R}^\delta(x)[2] &= \sigma\left(a\mathcal{R}^\delta(x)[1] + \delta x[2] + z_0\right) &(10)\\ &= \sigma\left(a\left(\sigma(z_0) + \delta\sigma'(z_0)x[1] + o(\delta)\right) + \delta x[2] + z_0\right), &(11)\end{aligned}$$

as the Taylor expansion at z_0 unless $a\sigma(z_0) = 0$.

In other words, the need for the additional width originates from that previous state $\mathcal{R}(x)[t-1]$ appears only after the second token. Nonetheless, we can make $\beta \equiv 0$, by setting the proper initial state for each RNN cell and using an affine projection map.

The lemma implies that a modified RNN can be approximated by an RNN with at most one additional width. For a given modified RNN $\bar{\mathcal{Q}} \circ \bar{\mathcal{R}}_L \circ \cdots \circ \bar{\mathcal{R}}_1 \circ \bar{\mathcal{P}}$ of width d and $\epsilon > 0$, we can find RNN $\mathcal{R}_1, \ldots, \mathcal{R}_{2L}$ and linear maps $\mathcal{P}_1, \ldots, \mathcal{P}_L$, $\mathcal{Q}_1, \ldots, \mathcal{Q}_L$ such that

$$\sup_{x\in K^N} \left\| \bar{\mathcal{Q}} \circ \bar{\mathcal{R}}_L \circ \cdots \circ \bar{\mathcal{R}}_1 \circ \bar{\mathcal{P}}(x) - (\mathcal{Q}_L\mathcal{R}_{2L}\mathcal{R}_{2L-1}\mathcal{P}_L) \circ \cdots \circ (\mathcal{Q}_1\mathcal{R}_2\mathcal{R}_1\mathcal{P}_1)(x) \right\| < \epsilon. \tag{12}$$

The composition $\mathcal{R} \circ \mathcal{P}$ of an RNN cell $\mathcal{R}$ and token-wise linear map $\mathcal{P}$ can be substituted by another RNN cell $\mathcal{R}'$. More concretely, for $\mathcal{R}$ and $\mathcal{P}$ defined by

$$\begin{aligned}\mathcal{R}(x)[t+1] &= \sigma\left(A\mathcal{R}(x)[t] + Bx[t+1] + \theta\right), &(13)\\ \mathcal{P}(x)[t] &= P\left(x[t]\right), &(14)\end{aligned}$$

$\mathcal{R} \circ \mathcal{P}$ defines an RNN cell $\mathcal{R}'$

$$\mathcal{R}'(x)[t+1] = \sigma\left(A\mathcal{R}'(x)[t] + BPx[t+1] + \theta\right). \tag{15}$$

Thus, $\mathcal{R}_{2l+1}(\mathcal{P}_{l+1}\mathcal{Q}_l)$ becomes a recurrent cell, and $(\mathcal{Q}_L\mathcal{R}_{2L}\mathcal{R}_{2L-1}\mathcal{P}_L)\circ\cdots\circ(\mathcal{Q}_1\mathcal{R}_2\mathcal{R}_1\mathcal{P}_1)(x)$ defines a network of form (5).

4. Universal Approximation for Deep RNN in Continuous Function Space

This section introduces the universal approximation theorem of deep RNNs in continuous function space.

Theorem 5 (Universal approximation theorem of deep RNN 1) *Let $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ be a continuous past-dependent sequence-to-sequence function and σ be a non-degenerate activation function. Then, for any $\epsilon > 0$ and compact subset $K \subset \mathbb{R}^{d_x}$, there exists a deep RNN $\mathcal{N}$ of width $d_x + d_y + 3 + \alpha(\sigma)$ such that*

$$\sup_{x \in K^N} \sup_{1 \le t \le N} \|f(x)[t] - \mathcal{N}(x)[t]\| < \epsilon, \tag{16}$$

where $\alpha(\sigma) = \begin{cases} 0 & \sigma \text{ is ReLU or a non-degenerating function with } \sigma(z_0) = 0. \\ 1 & \sigma \text{ is a non-degenerating function with } \sigma(z_0) \neq 0. \end{cases}$

To prove the above theorem, we deal with the case of the sequence-to-vector function $\mathcal{N} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y}$ first, in subsection 4. Then, we extend our idea to a sequence-to-sequence function using bias terms to separate the input vectors at different times in subsection 4, and the proof of Theorem 5 is presented at the end of this section. As the number of additional components required in each step in the subsections depends on the activation function, we use $\alpha(\sigma)$ to state the theorem briefly.

4.1 Approximation of sequence-to-vector function

The motivation for approximating a sequence-to-vector function $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y}$ is to copy a two-layered MLP via a modified RNN. Note that the output of a two-layered MLP is a linear sum of outputs of its hidden nodes, and each output of the hidden nodes is a linear sum of inputs with the following activation function. In Lemma 6, we construct a modified RNN that simulates a hidden node of an MLP, which can be represented as the sum of the inner product of some matrices and N input vectors in $\mathbb{R}^{d_x}$ with activation. After that, we use an additional buffer component in Lemma 7 to copy another hidden node in the two-layered MLP and the following modified RNN cell accumulates two results from the nodes. The buffer component of the modified RNN cell is then reset to zero to copy another hidden node. Repeating the procedure, the modified RNN with bounded width copies the two-layered MLP.

Now, we present the statements and sketches of the proof corresponding to each step. The following lemma implies that a modified RNN can compute the linear sum of all the input components, which copies the hidden node of a two-layered MLP.

Lemma 6 *Suppose $A[1], A[2], \cdots, A[N] \in \mathbb{R}^{1 \times d_x}$ are the given matrices. Then there exists a modified RNN $\mathcal{N} = \mathcal{R}_N \circ \mathcal{R}_{N-1} \circ \cdots \circ \mathcal{R}_1 \circ \mathcal{P} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{(d_x+1) \times N}$ of width $d_x + 1$ such that (the symbol $*$ indicates that there exists some value irrelevant to the proof)*

$$\begin{aligned} \mathcal{N}(x)[t] &= \begin{bmatrix} x[t] \\ * \end{bmatrix} && \text{for } t < N, \\ \mathcal{N}(x)[N] &= \begin{bmatrix} x[N] \\ \sigma\left(\sum_{t=1}^{N} A[t]x[t]\right) \end{bmatrix}. \end{aligned} \tag{17}$$

Proof [Sketch of the proof] The detailed proof is available in Appendix C. Define the m-th modified RNN cell $\mathcal{R}_m$, of the form of (1) without activation, with $A_m = \begin{bmatrix} O_{d_x \times d_x} & O_{d_x \times 1} \\ O_{1 \times d_x} & 1 \end{bmatrix}$, $B_m = \begin{bmatrix} I_{d_x} & O_{d_x \times 1} \\ b_m & 1 \end{bmatrix}$ where $b_m \in \mathbb{R}^{1 \times b_x}$. Then, the (d_x+1)th component $y[N]_{d_x+1}$ of the final output $y[N]$ after N layers becomes a linear combination of $b_i x[j]$ with some constant coefficients $\alpha_{i,j}$ and $\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{i,j} b_i x[j]$. Thus the coefficient of $x[j]$ is represented by $\sum_{i=1}^{N} \alpha_{i,j} b_i$, which we wish to be $A[j]$ for each $j = 1, 2, \ldots, N$. In matrix formulation, we intend to find b satisfying $\Lambda^T b = A$, where $\Lambda = \{\alpha_{i,j}\}_{1 \le i,j \le N} \in \mathbb{R}^{N \times N}$, $b = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} \in \mathbb{R}^{N \times d_x}$, and $A = \begin{bmatrix} A[1] \\ \vdots \\ A[N] \end{bmatrix}$. As Λ is invertible there exist b_i that solve $\left(\Lambda^T b\right)_j = A[j]$. ■

After copying a hidden node using the above lemma, we add a component, (d_x+2)th, to copy another hidden node. Then the results are accumulated in the (d_x+1)th component, and the final component is to be reset to copy another node. As the process is repeated, a modified RNN replicates the output node of a two-layered MLP.

Lemma 7 *Suppose $w_i \in \mathbb{R}$, $A_i[t] \in \mathbb{R}^{1 \times d_x}$ are given for $t = 1, 2, \ldots, N$ and $i = 1, 2, \ldots, M$. Then, there exists a modified RNN $\mathcal{N} : \mathbb{R}^{d_x \times N} \to \mathbb{R}$ of width $d_x + 2$ such that*

$$\mathcal{N}(x) = \sum_{i=1}^{M} w_i \sigma \left(\sum_{t=1}^{N} A_i[t] x[t] \right). \tag{18}$$

Proof First construct a modified RNN $\mathcal{N}_1 : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{(d_x+2) \times N}$ of width $d_x + 2$ such that

$$\mathcal{N}_1(x)[t] = \begin{bmatrix} x[t] \\ * \\ 0 \end{bmatrix} \qquad \text{for } t < N, \tag{19}$$

$$\mathcal{N}_1(x)[N] = \begin{bmatrix} x[N] \\ \sigma\left(\sum_{t=1}^{N} A_1[t] x[t] \right) \\ 0 \end{bmatrix}, \tag{20}$$

as Lemma 6. Note that the final component does not affect the first linear summation and remains zero. Next, using the components except for the (d_x+1)th one, construct $\mathcal{N}_2 : \mathbb{R}^{(d_x+2) \times N} \to \mathbb{R}^{(d_x+2) \times N}$, which satisfies

$$\mathcal{N}_2 \mathcal{N}_1(x)[t] = \begin{bmatrix} x[t] \\ * \\ * \end{bmatrix} \qquad \text{for } t < N, \tag{21}$$

$$\mathcal{N}_2 \mathcal{N}_1(x)[N] = \begin{bmatrix} x[N] \\ \sigma\left(\sum_{t=1}^{N} A_1[t] x[t] \right) \\ \sigma\left(\sum_{t=1}^{N} A_2[t] x[t] \right) \end{bmatrix}, \tag{22}$$

and use one modified RNN cell $\mathcal{R}$ after $\mathcal{N}_2$ to add the results and reset the last component:

$$\mathcal{R}\mathcal{N}_2\mathcal{N}_1(x)[t] = \begin{bmatrix} x[t] \\ * \\ 0 \end{bmatrix}, \tag{23}$$

$$\mathcal{R}\mathcal{N}_2\mathcal{N}_1(x)[N] = \begin{bmatrix} x[N] \\ w_1\sigma\left(\sum A_1[t]x[t]\right) + w_2\sigma\left(\sum A_2[t]x[t]\right) \\ 0 \end{bmatrix}. \tag{24}$$

As the (d_x+2)th component is reset to zero, we use it to compute the third sum $w_3\sigma\left(\sum A_3[t]x[t]\right)$ and repeat until we obtain the final network $\mathcal{N}$ such that

$$\mathcal{N}(x)[N] = \begin{bmatrix} x[N] \\ \sum_{i=1}^{M} w_i\sigma\left(\sum_{t=1}^{N} A_i[t]x[t]\right) \\ 0 \end{bmatrix}. \tag{25}$$

■

Remark 8 *The above lemma implies that a modified RNN of width $d_x + 2$ can copy the output node of a two-layered MLP. We can extend this result to an arbitrary d_y-dimensional case. Note that the first d_x components remain fixed, the $(d_x + 1)$th component computes a part of the linear sum approximating the target function, and the $(d_x + 2)$th component computes another part and is reset. When we need to copy another output node for another component of the output of the target function $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$, only one additional width is sufficient. Indeed, the (d_x+2)th component computes the sum and the final component, and the (d_x+3)th component acts as a buffer to be reset in that case. By repeating this process, we obtain $(d_x + d_y + 1)$-dimensional output from the modified RNN, which includes all d_y outputs of the MLP and the components from the $(d_x + 1)$th to the $(d_x + d_y)$th ones.*

Theorem 9 (Universal approximation theorem of deep RNN 2) *Suppose a target $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y}$ is a continuous sequence-to-vector function, $K \subset \mathbb{R}^{d_x}$ is a compact subset, σ is a non-degenerating activation function, and z_0 is the non-degenerating point. Then, for any $\epsilon > 0$, there exists a deep RNN $\mathcal{N} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y}$ of width $d_x + d_y + 1 + \beta(\sigma)$ such that*

$$\sup_{x \in K^N} \|f(x) - \mathcal{N}(x)\| < \epsilon, \tag{26}$$

where $\beta(\sigma) = \begin{cases} 0 & \text{if } \sigma(z_0) = 0 \\ 1 & \text{otherwise} \end{cases}$

Proof We present the proof for $d_y = 1$ here, but adding $d_y - 1$ width for each output component works for the case $d_y > 1$. By the universal approximation theorem of the MLP, Theorem 1 in Leshno et al. (1993), there exist w_i and $A_i[t]$ for $i = 1, \ldots, M$ such that

$$\sup_{x \in K^N} \left\| f(x) - \sum_{i=1}^{M} w_i\sigma\left(\sum_{t=1}^{N} A_i[t]x[t]\right) \right\| < \frac{\epsilon}{2}. \tag{27}$$

Note that there exists a modified RNN $\bar{\mathcal{N}} : \mathbb{R}^{d_x \times N} \to \mathbb{R}$ of width $d_x + 2$,

$$\bar{\mathcal{N}}(x) = \sum_{i=1}^{M} w_i \sigma \left(\sum_{t=1}^{N} A_i[t] x[t] \right). \tag{28}$$

By Lemma 3, there exists an RNN $\mathcal{N} : \mathbb{R}^{d_x \times N} \to \mathbb{R}$ of width $d_x + 2 + \beta(\sigma)$ such that

$$\sup_{x \in K^n} \left\| \bar{\mathcal{N}}(x) - \mathcal{N}(x) \right\| < \frac{\epsilon}{2}. \tag{29}$$

Hence we have $\|f(x) - \mathcal{N}(x)\| < \epsilon$. ■

4.2 Approximation of sequence-to-sequence function

Now, we consider an RNN $\mathcal{R}$ as a function from sequence x to sequence $y = \mathcal{R}(x)$ defined by (1). Although the above results are remarkable in that the minimal width has an upper bound independent of the length of the sequence, it only approximates a part of the output sequence. Meanwhile, as the hidden states calculated in each RNN cell are connected closely for different times, fitting all the functions that can be independent of each other becomes a more challenging problem. For example, the coefficient of $x[t-1]$ in $\mathcal{N}(x)[t]$ equals the coefficient of $x[t]$ in $\mathcal{N}(x)[t+1]$ if $\mathcal{N}$ is an RNN defined as in the proof of Lemma 6. This correlation originates from the fact that $x[t-1]$ and $x[t]$ arrive at $\mathcal{N}(x)[t], \mathcal{N}(x)[t+1]$ via the same intermediate process, 1-time step, and N layers.

We sever the correlation between the coefficients of $x[t-1]$ and $x[t]$ by defining the *time-enhanced recurrent cell* in Definition 10 and proceed similarly as in the previous subsection till the Lemma 12.

Definition 10 *Time-enhanced recurrent cell, or layer, is a process that maps sequence* $x = (x[t])_{t \in \mathbb{N}} \in \mathbb{R}^{d_s \times \mathbb{N}}$ *to sequence* $y = (y[t])_{t \in \mathbb{N}} \in \mathbb{R}^{d_s \times \mathbb{N}}$ *via*

$$y[t+1] \coloneqq \mathcal{R}(x)[t+1] = \sigma \left(A[t+1] \mathcal{R}(x)[t] + B[t+1] x[t+1] + \theta[t+1] \right) \tag{30}$$

where σ *is an activation function,* $A[t]$, $B[t] \in \mathbb{R}^{d_s \times d_s}$ *are weight matrices and* $\theta[t] \in \mathbb{R}^{d_s}$ *is the bias given for each time step* t.

Like RNN, *time-enhanced RNN* indicates a composition of the form (1) with time-enhanced recurrent cells instead of RNN cells, and we denote it as TRNN. The *modified TRNN* indicates a TRNN whose activation functions in some cell act on only part of the components. *Time-enhanced BRNN*, denoted as *TBRNN*, indicates a BRNN whose recurrent layers in each direction are replaced by time-enhanced layers. A *modified TBRNN* indicates a TBRNN whose activation function is modified to act on only part of the components. With the proof of Lemma 3 using $\bar{A}[t]$, $\bar{B}[t]$ instead of $\bar{A}$, $\bar{B}$, a TRNN can approximate a modified TRNN.

The following lemma shows that the modified TRNN successfully eliminates the correlation between outputs. See the appendix for the complete proof.

Lemma 11 *Suppose $A_j[t] \in \mathbb{R}^{1\times d_x}$ are the given matrices for $1 \le t \le N$, $1 \le j \le t$. Then there exists a modified TRNN $\tilde{\mathcal{N}} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{(d_x+1)\times N}$ of width $d_x + 1$ such that*

$$\tilde{\mathcal{N}}(x)[t] = \begin{bmatrix} x[t] \\ \sigma\left(\sum_{j=1}^{t} A_j[t]x[j]\right) \end{bmatrix}, \tag{31}$$

for all $t = 1, 2, \ldots, N$.

Proof [Sketch of proof] The detailed proof is available in Appendix D. The main idea is to use $b_m[t] \in \mathbb{R}^{1\times d_x}$ instead of $b_m \in \mathbb{R}^{1\times d_x}$ in the proof of Lemma 6 with a modified TRNN $\tilde{\mathcal{R}}$ of the form

$$\tilde{\mathcal{R}}(x)[t+1] \coloneqq \sigma_I\left(\begin{bmatrix} O_{d_x\times d_x} & \\ & 1 \end{bmatrix} \tilde{\mathcal{R}}(x)[t] + \begin{bmatrix} I_{d_x\times d_x} & 0 \\ b_m[t+1] & 1 \end{bmatrix} x[t+1]\right). \tag{32}$$

The time-index coefficient $b_m[t]$ separates the calculation along the same intermediate process mentioned at the beginning of this subsection. For instance, a process from time t_1 to $t_1 + 1$ results differently from t_2 to $t_2 + 1$ for $b_m[t_1] \neq b_m[t_2]$, even in the same layer. As the coefficient matrices at each time $[t]$ after N layers are full rank, we can find $b_m[t]$ implementing the required linear combination for each time. ■

Recall the proof of Lemma 7. An additional width serves as a buffer to implement and accumulate linear sum in a node in an MLP. Similarly, we proceed with Lemma 11 instead of Lemma 6 to conclude that there exists a modified TRNN $\mathcal{N}$ of width $d_x + 2$ such that each $\mathcal{N}[t]$ reproduces an MLP approximating $f[t]$.

Lemma 12 *Suppose $w_i[t] \in \mathbb{R}$, $A_{i,j}[t] \in \mathbb{R}^{1\times d_x}$ are the given matrices for $1 \le t \le N$, $1 \le j \le t$, $1 \le i \le M$. Then, there exists a modified TRNN $\tilde{\mathcal{N}} : \mathbb{R}^{d_x\times N} \to \mathbb{R}^{1\times N}$ of width $d_x + 2$ such that*

$$\tilde{\mathcal{N}}(x)[t] = \sum_{i=1}^{M} w_i[t]\sigma\left(\sum_{j=1}^{t} A_{i,j}[t]x[j]\right) \tag{33}$$

Proof We omit the detailed proof because it is almost the same as the proof of Lemma 7. The only difference is to use $b_m[t]$ instead of b_m to construct $A_{i,j}[t]$ and $w_i[t]$ as in the proof of Lemma 11. ■

This implies that the modified TRNN can approximate any past-dependent sequence-to-sequence function.

Finally, we connect the TRNN and RNN. Although it is unclear whether a modified RNN can approximate an arbitrary modified TRNN, there exists a modified RNN that approximates the specific one described in Lemma 11.

Lemma 13 *Let $\tilde{\mathcal{N}}$ be a given modified TRNN that computes (31) with width $d_x + 1$ and $K \subset \mathbb{R}^{d_x}$ be a compact set. Then, for any $\epsilon > 0$ there exists a modified RNN $\mathcal{N}$ of width $d_x + 2 + \gamma(\sigma)$ such that*

$$\sup_{x\in K^N} \left\| \tilde{\mathcal{N}}(x) - \mathcal{N}(x) \right\| < \epsilon, \tag{34}$$

where $\gamma(\mathrm{ReLU}) = 0$, $\gamma(\sigma) = 1$ for non-degenerating activation σ. As a corollary, there exists a modified RNN $\mathcal{N}$ of width $d_x + 3 + \gamma(\sigma)$ approximating $\tilde{\mathcal{N}}$ in Lemma 12.

Proof [Sketch of proof] Detailed proof is available in Appendix E. Note that a function $\tilde{\mathcal{N}}$ in (31) is constructed by modified TRNN cells of the form in (32). Since $b_m[t]$ is the only time-dependent coefficient in (32), it is enough to show that an RNN cell can approximate a function $X \to \begin{bmatrix} I_{d_x} & O_{d_x \times 1} \\ b_m[t] & 1 \end{bmatrix} X$ for $X = \begin{bmatrix} x[t] \\ 0 \end{bmatrix} \in \mathbb{R}^{d_x+1}$. In particular, approximating a map $x[t] \mapsto b_m[t]x[t]$ is what we need.

While an MLP can approximate map $x[t] \mapsto b_m[t]x[t]$ on a compact set K, it is hard to use token-wise MLP since we need different outputs for the same vector at other time steps. For instance, for a $v \in K$ and a sequence $x = (x[1], x[2]) = (v, v) \in K^2$, a token-wise MLP cannot simultaneously approximate two outputs, $b_m[1]v$ and $b_m[2]v$. Still, it is not the case when the domain K_1 of $x[1]$ and the domain K_2 of $x[2]$ are disjoint compact sets, and there is an MLP that approximates $v \mapsto b_m[1]v$ for $v \in K_1$ and $v \mapsto b_m[2]v$ for $v \in K_2$. From this point of view, we use an RNN and a token-wise MLP, to embed each vector of the input sequence into disjoint compact sets and to approximate the output $b_m[t]x[t]$ on each compact set, respectively.

Now we present how to construct disjoint sets and the token-wise MLP.

Without loss of generality, we can assume $K \subset \left[0, \frac{1}{2}\right]^{d_x}$ and construct the first cell as the output at time t to be $x[t] + t\mathbf{1}_{d_x}$. As N compact sets $K + \mathbf{1}_{d_x}, K + 2\mathbf{1}_{d_x}, \ldots, K + N\mathbf{1}_{d_x}$ are disjoint, t and $x[t] = y - t$ are uniquely determined for each $y \in K + t\mathbf{1}_{d_x}$. Thus, for any given $b[t] \in \mathbb{R}^{1 \times d_x}$, there exists an MLP of width $d_x + 1 + \gamma(\sigma)$ approximating $y = x[t] + t\mathbf{1}_{d_x} \mapsto b[t]x[t]$ on $\bigsqcup_t (K + t\mathbf{1}_{d_x})$ as a function from $\mathbb{R}^{d_x} \to \mathbb{R}$ (Hanin and Sellke, 2017; Kidger and Lyons, 2020). Indeed, we need to approximate $x[t] + t\mathbf{1}_{d_x} \mapsto \begin{bmatrix} x[t] + t\mathbf{1}_{d_x} \\ b[t]x[t] \end{bmatrix}$ as a function from $\mathbb{R}^{d_x}$ to $\mathbb{R}^{d_x+1}$. Fortunately, the first d_x components preserve the original input data in the proofs of **Proposition 2** in Hanin and Sellke (2017), and **Proposition 4.2(Register Model)** in Kidger and Lyons (2020). Thus, an MLP of width $d_x + 1 + \gamma(\sigma)$ approximates $b[t]x[t]$ while keeping the $x + t\mathbf{1}_{d_x}$ in the first d_x components, and the MLP is common across overall t as inputs at different times t_1 and t_2 are already embedded in disjoint sets $K + t_1\mathbf{1}_{d_x}$ and $K + t_2\mathbf{1}_{d_x}$. Note that a token-wise MLP is a special case of an RNN of the same width. Nonetheless, we need an additional width to keep the $(d_x + 1)$th component approximating $b[t]x[t]$. With the token-wise MLP implemented by an RNN and additional buffer width, we construct a modified RNN of width $d_x + 2 + \gamma(\sigma)$ approximating the modified TRNN cell used in the proof of Lemma 11. ■

4.3 Proof of Theorem 5

Summarizing all the results, we have the universality of a deep RNN in a continuous function space.

Proof [Proof of Theorem 5] As mentioned in Remark 8, we can set $d_y = 1$ for notational convenience. By Lemma 12, there exists a modified TRNN $\tilde{\mathcal{N}}$ of width $d_x + 2$ such that

$$\sup_{x \in K^n} \left\| f(x) - \tilde{\mathcal{N}}(x) \right\| < \frac{\epsilon}{3}. \tag{35}$$

As $\tilde{\mathcal{N}}$ is a composition of modified TRNN cells of width $d_x + 2$ satisfying (31), there exists a modified RNN $\bar{\mathcal{N}}$ of width $d_x + 3 + \gamma(\sigma)$ such that

$$\sup_{x \in K^n} \left\| \tilde{\mathcal{N}}(x) - \bar{\mathcal{N}}(x) \right\| < \frac{\epsilon}{3}. \tag{36}$$

Then, by Lemma 3, there exists an RNN $\mathcal{N}$ of width $d_x + 3 + \gamma(\sigma) + \beta(\sigma) = d_x + 4 + \alpha(\sigma)$ such that

$$\sup_{x \in K^n} \left\| \bar{\mathcal{N}}(x) - \mathcal{N}(x) \right\| < \frac{\epsilon}{3}. \tag{37}$$

The triangle inequality yields

$$\sup_{x \in K^n} \| f(x) - \mathcal{N}(x) \| < \epsilon. \tag{38}$$

■

Remark 14 *The number of additional widths $\alpha(\sigma) = \beta(\sigma) + \gamma(\sigma)$ depends on the condition of the activation function σ. Here, $\gamma(\sigma)$ is required to find the token-wise MLP that approximates embedding from $\mathbb{R}^{d_x}$ to $\mathbb{R}^{d_x+1}$. If further studies determine a tighter upper bound of the minimum width of an MLP to have the universal property in a continuous function space, we can reduce or even remove $\alpha(\sigma)$ according to the result.*

There is still a wide gap between the lower bound d_x and upper bound $d_x + d_y + 4 + \alpha(\sigma)$ of the minimum width, and hence, we expect to be able to achieve universality with a narrower width. For example, if $N = 1$, an RNN is simply an MLP, and the RNN has universality without a node required to compute the effect of t. Therefore, apart from the result of the minimum width of an MLP, further studies are required to determine whether γ is essential for the case of $N \geq 2$.

5. Universal Approximation for Stack RNN in L^p Space

This section introduces the universal approximation theorem of a deep RNN in L^p function space for $1 \leq p < \infty$.

Theorem 15 (Universal approximation theorem of deep RNN 3) *Let $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ be a past-dependent sequence-to-sequence function in $L^p\left(\mathbb{R}^{d_x \times N}, \mathbb{R}^{d_y \times N}\right)$ for $1 \leq p < \infty$, and σ be a non-degenerate activation function with the non-degenerating point z_0. Then, for any $\epsilon > 0$ and compact subset $K \subset \mathbb{R}^{d_x}$, there exists a deep RNN $\mathcal{N}$ of width $\max\{d_x + 1, d_y\} + 1$ satisfying*

$$\sup_{1 \leq t \leq N} \| f(x)[t] - \mathcal{N}(x)[t] \|_{L^p(K)} < \epsilon. \tag{39}$$

Moreover, if the activation σ is ReLU, there exists a deep RNN $\mathcal{N}$ of width $\max\{d_x + 1, d_y\}$ satisfying

$$\sup_{1 \leq t \leq N} \| f(x)[t] - \mathcal{N}(x)[t] \|_{L^p\left(\mathbb{R}^{d_x}\right)} < \epsilon. \tag{40}$$

Before beginning the proof of the theorem, we assume $K \subseteq [0,1]^{d_x}$ without loss of generality and summarize the scheme that will be used in the proof. Park et al. (2021) constructed an MLP of width $\max\{d_x+1, d_y\}+\gamma(\sigma)$ approximating a given target function f, using the "encoding scheme." More concretely, the MLP is separated into three parts: encoder, memorizer, and decoder.

First, the encoder part quantizes each component of the input and output into a finite set. The authors use the quantization function $q_n : [0,1] \to \mathcal{C}_n$

$$q_n(v) \coloneqq \max\left\{c \in \mathcal{C}_n \;\middle|\; c \le v\right\}, \tag{41}$$

where $\mathcal{C}_n \coloneqq \{0, 2^{-n}, 2\times 2^{-n}, \dots, 1-2^{-n}\}$. Then, each quantized vector is encoded into a real number by concatenating its components through the encoder $\mathrm{Enc}_M : [0,1]^{d_x} \to \mathcal{C}_{d_x M}$

$$\mathrm{Enc}_M(x) \coloneqq \sum_{i=1}^{d_x} q_M(x_i) 2^{-(i-1)M}. \tag{42}$$

For small $\delta_1 > 0$, the authors construct an MLP $\mathcal{N}_{enc} : [0,1]^{d_x} \to \mathcal{C}_{d_x M}$ of width $d_x+1+\gamma(\sigma)$ satisfying

$$\|\mathrm{Enc}_M(x) - \mathcal{N}_{enc}(x)\| < \delta_1. \tag{43}$$

Although the quantization causes a loss in input information, the L^p norm neglects some loss in a sufficiently small domain.

After encoding the input x to $\mathrm{Enc}_M(x)$ with large M, authors use the information of x in $\mathrm{Enc}_M(x)$ to obtain the information of the target output $f(x)$. More precisely, they define the memorizer $\mathrm{Mem}_{M,M'} : \mathcal{C}_{d_x M} \to \mathcal{C}_{d_y M'}$ to map the encoded input $\mathrm{Enc}_M(x)$ to the encoded output $\mathrm{Enc}_{M'}(f(x))$ as

$$\mathrm{Mem}\left(\mathrm{Enc}_M(x)\right) \coloneqq \left(\mathrm{Enc}_{M'} \circ f \circ q_M\right)(x), \tag{44}$$

assuming the quantized map q_M acts on x component-wise in the above equation. Then, an MLP $\mathcal{N}_{mem}$ of width $2+\gamma(\sigma)$ approximates Mem; that is, for any $\delta_2 > 0$, there exists $\mathcal{N}_{mem}$ satisfying

$$\sup_{x\in[0,1]^{d_x}} \|\mathrm{Mem}(\mathrm{Enc}(x)) - \mathcal{N}_{mem}(\mathrm{Enc}(x))\| < \delta_2. \tag{45}$$

Finally, the decoder reconstructs the original output vector from the encoded output vector by cutting off the concatenated components. Owing to the preceding encoder and memorizer, it is enough to define only the value of the decoder on $\mathcal{C}_{d_y M'}$. Hence the decoder $\mathrm{Dec} : \mathcal{C}_{d_y M'} \to \mathcal{C}_{M'}^{d_y} \coloneqq (\mathcal{C}_{M'})^{d_y}$ is determined by

$$\mathrm{Dec}_{M'}(v) \coloneqq \hat{v} \qquad \text{where} \qquad \{\hat{v}\} \coloneqq \mathrm{Enc}_{M'}^{-1}(v) \cap \mathcal{C}_{M'}^{d_y}. \tag{46}$$

Indeed, for small $\delta_3 > 0$, Park et al. (2021) construct an MLP $\mathcal{N}_{dec} : \mathcal{C}_{d_y M'} \to \mathcal{C}_{M'}^{d_y}$ of width $d_y+\gamma(\sigma)$ so that

$$\|\mathrm{Dec}_{M'}(v) - \mathcal{N}_{dec}(v)\| < \delta_3 \tag{47}$$

Although (43) and (47) are not equations but approximations when the activation is just non-degenerate, the composition $\mathcal{N} = \mathcal{N}_{dec} \circ \mathcal{N}_{mem} \circ \mathcal{N}_{enc}$ approximates a target f with sufficiently large M, M' and sufficiently small δ_1, δ_2.

Let us return to the proof of Theorem 15. We construct the encoder, memorizer, and decoder similarly. As the encoder and decoder is independent of time t, we use a token-wise MLP and modified RNNs define the token-wise MLPs. On the other hand, the memorizer must work differently according to the time t owing to the multiple output functions. Instead of implementing various memorizers, we separate their input and output domains at each time by translation. Then, it is enough to define one memorizer on the disjoint union of domains.

Proof [Proof of Theorem 15] We first combine the token-wise encoder and translation for the separation of the domains. Consider the token-wise encoder $\mathrm{Enc}_M : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{1\times N}$, and the following recurrent cell $\mathcal{R} : \mathbb{R}^{1\times N} \to \mathbb{R}^{1\times N}$

$$\mathcal{R}(v)[t+1] = 2^{-(d_x M+1)}\mathcal{R}(v)[t] + v[t+1] + 1. \tag{48}$$

Then the composition $\mathcal{R}_{enc} = \mathcal{R}\,\mathrm{Enc}_M$ defines an encoder of sequence from K^N to $\mathbb{R}^{1\times N}$:

$$\mathcal{R}_{enc}(x)[t] = \sum_{j=1}^{t} 2^{-(j-1)d_x M} + \sum_{j=1}^{t} \mathrm{Enc}_M\left(x[j]\right) 2^{-(j-1)(d_x M+1)}, \tag{49}$$

where $x = (x[t])_{t=1,\dots,N}$ is a sequence in K. Note that the range D of $\mathcal{R}_{enc}$ is a disjoint union of compact sets;

$$D = \bigsqcup_{t=1}^{N} \left\{\mathcal{R}_{enc}(x)[t] : x \in K^N\right\}. \tag{50}$$

Hence there exists a memorizer $\mathrm{Mem} : \mathbb{R} \to \mathbb{R}$ satisfying

$$\mathrm{Mem}(\mathcal{R}_{enc}(x)[t]) = \mathrm{Enc}_{M'}\left(f\left(q_M(x)\right)[t]\right) \tag{51}$$

for each $t = 1, 2, \dots, N$. The token-wise decoder $\mathrm{Dec}_{M'}$ is the last part of the proof.

To complete the proof, we need an approximation of the token-wise encoder $\mathrm{Enc}_M : \mathbb{R}^{d_x} \to \mathbb{R}$, a modified recurrent cell $\mathcal{R} : \mathbb{R}^{1\times N} \to \mathbb{R}^{1\times N}$, token-wise memorizer $\mathrm{Mem} : \mathbb{R} \to \mathbb{R}$, and token-wise decoder $\mathrm{Dec}_{M'} : \mathbb{R} \to \mathbb{R}^{d_y}$. Following Park et al. (2021), there exist MLPs of width $d_x + 1 + \gamma(\sigma)$, $2 + \gamma(\sigma)$, and $d_y + \gamma(\sigma)$ that approximate Enc_M, Mem, and $\mathrm{Dec}_{M'}$ respectively. Lemma 3 shows that $\mathcal{R}$ is approximated by an RNN of width $1 + \beta(\sigma)$. Hence, an RNN of width $\max\{d_x + 1 + \gamma(\sigma), 1 + \beta(\sigma), 2 + \gamma(\sigma), d_y + \gamma(\sigma)\} = \max\{d_x + 1, d_y\} + \gamma(\sigma)$ approximates the target function f.

In the case of ReLU activation, we can extend the domain K^N to $\mathbb{R}^{d_x\times N}$ as stated in Park et al. (2020). Nonetheless, we present briefly how to deal with the subtle problem that the support of a network is not compact generally.

We project each input $x[t] \in \mathbb{R}^{d_x}$ by $P_{L,\delta} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ defined by

$$P(x) = \begin{cases} x & x \in [-L, L], \\ -\frac{L}{\delta}\left(x + L + \delta\right) & x \in [-L-\delta, -L], \\ -\frac{L}{\delta}\left(x - L - \delta\right) & x \in [L, L+\delta], \\ 0 & \text{otherwise.} \end{cases} \tag{52}$$

Note that an MLP with width $d_x + 1$ computes $P_{L,\delta}$. We will choose large L to cover enough domain for $x \in \mathbb{R}^{d_x\times N}$ and small δ to reduce the projection error.

First, choose a large L so that $\|f(x)[t]\|_{L^p\left(\mathbb{R}^{d_x}\setminus[-L,L]^{d_x}\right)} < \frac{\epsilon}{3}$ for all $t = 1, 2, \ldots, N$. Second, construct an RNN $\mathcal{N} = \mathcal{N}_{dec} \circ \mathcal{N}_{mem} \circ \mathcal{N}_{enc}$ as above with $\mathcal{N}(0) = 0$, such that $\|f(x)[t] - \mathcal{N}(x)[t]\|_{L^p\left([-L,L]^{d_x}\right)} < \frac{\epsilon}{3}$. We impose a condition $\mathcal{N}(0) = 0$ to bound error outside $[-L, L]^{d_x}$, which is possible because we may set $f(0)[t] = 0$ for all t under the L^p norm. Finally, set δ so small that $\|f(x)[t]\|_{L^p\left([-L-\delta,L+\delta]^{d_x}\setminus[-L,L]^{d_x}\right)} < \frac{\epsilon}{6}$ and $\|\mathcal{N}(x)[t]\|_{L^p\left([-L-\delta,L+\delta]^{d_x}\setminus[-L,L]^{d_x}\right)} < \frac{\epsilon}{6}$. After inserting the projection $P_{L,\delta}$ before the encoder $\mathcal{N}_{enc}$ and defining new RNN $\mathcal{N}' = \mathcal{N} \circ P_{L,\delta}$, we have

$$
\begin{aligned}
&\left\|f(x)[t] - \mathcal{N}'(x)[t]\right\|_{L^p\left(\mathbb{R}^{d_x}\right)} && (53)\\
&\leq \|f(x)[t] - \mathcal{N}(x)[t]\|_{L^p\left([-L,L]^{d_x}\right)} + \left\|f(x)[t] - \mathcal{N}'(x)[t]\right\|_{L^p\left(\mathbb{R}^{d_x}\setminus[-L,L]^{d_x}\right)} && (54)\\
&\leq \frac{\epsilon}{3} + \|f(x)[t] - \mathcal{N}(x)[t]\|_{L^p\left([-L-\delta,L+\delta]^{d_x}\setminus[-L,L]^{d_x}\right)} && (55)\\
&\quad + \|f(x)[t] - \mathcal{N}(x)[t]\|_{L^p\left(\mathbb{R}^{d_x}\setminus[-L-\delta,L+\delta]^{d_x}\right)} && (56)\\
&\leq \frac{\epsilon}{3} + \frac{\epsilon}{3} + \|f(x)[t]\|_{L^p\left(\mathbb{R}^{d_x}\setminus[-L,L]^{d_x}\right)} && (57)\\
&\leq \epsilon && (58)
\end{aligned}
$$

■

6. Variants of RNN

This section describes the universal property of some variants of RNN, particularly LSTM, GRU, or BRNN. LSTM and GRU are proposed to solve the long-term dependency problem. As an RNN has difficulty calculating and updating its parameters for long sequential data, LSTM and GRU take advantage of additional structures in their cells. We prove that they have the same universal property as the original RNN. On the other hand, a BRNN is proposed to overcome the past dependency of an RNN. BRNN consists of two RNN cells, one of which works in reverse order. We prove the universal approximation theorem of a BRNN with the target class of any sequence-to-sequence function.

The universal property of an LSTM originates from the universality of an RNN. Mathematically LSTM $\mathcal{R}_{LSTM}$ indicates a process that computes two outputs, h and c, defined by (2). As an LSTM can reproduce an RNN with the same width, we have the following corollary:

Corollary 16 (Universal approximation theorem of deep LSTM) *Let $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ be a continuous past-dependent sequence-to-sequence function. Then, for any $\epsilon > 0$ and compact subset $K \subset \mathbb{R}^{d_x}$, there exists a deep LSTM $\mathcal{N}_{LSTM}$, of width $d_x + d_y + 3$, such that*

$$\sup_{x\in K^N} \sup_{1\leq t\leq N} \|f(x)[t] - \mathcal{N}_{LSTM}(x)[t]\| < \epsilon. \tag{59}$$

If $f \in L^p\left(\mathbb{R}^{d_x\times N}, \mathbb{R}^{d_y\times N}\right)$, there exists a deep LSTM $\mathcal{N}_{LSTM}$, of width $\max\{d_x + 1, d_y\}+1$, such that

$$\sup_{1\leq t\leq N} \|f(x)[t] - \mathcal{N}_{LSTM}(x)[t]\|_{L^p(K^N)} < \epsilon. \tag{60}$$

Proof We set all parameters but W_g, U_g, b_g, and b_f as zeros, and then (2) is simplified as

$$\begin{aligned} c[t+1] &= \sigma_{\text{sig}}(b_f) \odot c[t] + \frac{1}{2}\tanh\left(U_g h[t] + W_g x[t+1] + b_g\right), \\ h[t+1] &= \frac{1}{2}\tanh\left(c[t+1]\right). \end{aligned} \tag{61}$$

For any $\epsilon > 0$, b_f with sufficiently large negative components yields

$$\left\| h[t+1] - \frac{1}{2}\tanh\left(\frac{1}{2}\tanh\left(U_g h[t] + W_g x[t+1] + b_g\right)\right)\right\| < \epsilon. \tag{62}$$

Thus, an LSTM reproduces an RNN whose activation function is $\left(\frac{1}{2}\tanh\right)\circ\left(\frac{1}{2}\tanh\right)$ without any additional width in its hidden states. In other words, an LSTM of width d approximates an RNN of width d equipped with the activation function $\left(\frac{1}{2}\tanh\right)\circ\left(\frac{1}{2}\tanh\right)$. ■

The universality of GRU is proved similarly.

Corollary 17 (Universal approximation theorem of deep GRU) *Let $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ be a continuous past-dependent sequence-to-sequence function. Then, for any $\epsilon > 0$ and compact subset $K \subset \mathbb{R}^{d_x}$, there exists a deep GRU $\mathcal{N}_{GRU}$, of width $d_x + d_y + 3$, such that*

$$\sup_{x \in K^N} \sup_{1 \le t \le N} \|f(x)[t] - \mathcal{N}_{GRU}(x)[t]\| < \epsilon. \tag{63}$$

If $f \in L^p\left(\mathbb{R}^{d_x \times N}, \mathbb{R}^{d_y \times N}\right)$, there exists a deep GRU $\mathcal{N}_{GRU}$, of width $\max\{d_x + 1, d_y\} + 1$, such that

$$\sup_{1 \le t \le N} \|f(x)[t] - \mathcal{N}_{GRU}(x)[t]\|_{L^p(K^N)} < \epsilon. \tag{64}$$

Proof Setting only W_h, U_h, b_h, and b_z as non-zero, the GRU is simplified as

$$h[t+1] = \left(1 - \sigma_{\text{sig}}\left(b_z\right)\right) h[t] + \sigma_{\text{sig}}\left(b_z\right) \tanh\left(W_h x[t+1] + \frac{1}{2} U_h h[t] + b_h\right). \tag{65}$$

For any $\epsilon > 0$, a sufficiently large b_z yields

$$\left\| h[t+1] - \tanh\left(W_h x[t+1] + \frac{1}{2} U_h h[t] + b_h\right)\right\| < \epsilon. \tag{66}$$

Hence, we attain the corollary. ■

Remark 18 *We refer to the width as the maximum of hidden states. However, the definition is somewhat inappropriate, as LSTM and GRU cells have multiple hidden states; hence, there are several times more components than an RNN with the same width. Thus we expect that they have better approximation power or have a smaller minimum width for universality than an RNN. Nevertheless, we retain the theoretical proof as future work to identify whether they have different abilities in approximation or examine why they exhibit different performances in practical applications.*

Now, let us focus on the universality of a BRNN. Recall that a stack of modified recurrent cells $\mathcal{N}$ construct a linear combination of the previous input components $x[1:t]$ at each time,

$$\mathcal{N}(x)[t] = \begin{bmatrix} x[t] \\ \sum_{j=1}^{t} A_j[t]x[j] \end{bmatrix}. \tag{67}$$

Therefore, if we reverse the order of sequence and flow of the recurrent structure, a stack of reverse modified recurrent cells $\bar{\mathcal{N}}$ constructs a linear combination of the subsequent input components $x[t:N]$ at each time,

$$\bar{\mathcal{N}}(x)[t] = \begin{bmatrix} x[t] \\ \sum_{j=t}^{N} B_j[t]x[j] \end{bmatrix}. \tag{68}$$

From this point of view, we expect that a stacked BRNN successfully approximates an arbitrary sequence-to-sequence function beyond the past dependency. As previously mentioned, we prove it in the following lemma.

Lemma 19 *Suppose $A_j[t] \in \mathbb{R}^{1\times d_x}$ are the given matrices for $1 \le t \le N$, $1 \le j \le N$. Then there exists a modified TBRNN $\tilde{\mathcal{N}} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{(d_x+1)\times N}$ of width $d_x + 1$ such that*

$$\tilde{\mathcal{N}}(x)[t] = \begin{bmatrix} x[t] \\ \sigma\left(\sum_{j=1}^{N} A_j[t]x[j]\right) \end{bmatrix}, \tag{69}$$

for all $t = 1, 2, \ldots, N$.

Proof [Sketch of proof] The detailed proof is available in Appendix F. We use modified TBRNN cells with either only a forward modified TRNN or a backward modified TRNN. The stacked forward modified TRNN cells compute $\sum_{j=1}^{t} A_j[t]x[j]$, and the stacked backward modified TRNN cells compute $\sum_{j=t+1}^{N} A_j[t]x[j]$. ■

As in previous cases, we have the following theorem for a TBRNN. The proof is almost the same as that of Lemma 12 and 7.

Lemma 20 *Suppose $w_i[t] \in \mathbb{R}$, $A_{i,j}[t] \in \mathbb{R}^{1\times d_x}$ are the given matrices for $1 \le t \le N$, $1 \le j \le N$, $1 \le i \le M$. Then there exists a modified TBRNN $\tilde{\mathcal{N}} : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{1\times N}$ of width $d_x + 2$ such that*

$$\tilde{\mathcal{N}}(x)[t] = \sum_{i=1}^{M} w_i[t]\sigma\left(\sum_{j=1}^{N} A_{i,j}[t]x[j]\right). \tag{70}$$

Proof First, construct a modified deep TBRNN $\mathcal{N}_1 : \mathbb{R}^{d_x\times N} \to \mathbb{R}^{(d_x+2)\times N}$ of width $d_x + 2$ such that

$$\mathcal{N}_1(x)[t] = \begin{bmatrix} x[t] \\ \sigma\left(\sum_{j=1}^{N} A_{1,j}[t]x[j]\right) \\ 0 \end{bmatrix}, \tag{71}$$

as Lemma 19. The final component does not affect the first linear summation and remains zero. After $\mathcal{N}_1$, use the (d_x+2)th component to obtain a stack of cells $\mathcal{N}_2 : \mathbb{R}^{(d_x+2)\times N} \to \mathbb{R}^{(d_x+2)\times N}$, which satisfies

$$\mathcal{N}_2\mathcal{N}_1(x)[t] = \begin{bmatrix} x[t] \\ \sigma\left(\sum_{j=1}^{N} A_{1,j}[t]x[j]\right) \\ \sigma\left(\sum_{j=1}^{N} A_{2,j}[t]x[j]\right) \end{bmatrix}, \tag{72}$$

and use a modified RNN cell $\mathcal{R}$ to sum up the results and reset the last component:

$$\mathcal{R}\mathcal{N}_2\mathcal{N}_1(x)[t] = \begin{bmatrix} x[t] \\ w_1[t]\sigma\left(\sum_{j=1}^{N} A_{1,j}[t]x[j]\right) + w_2[t]\sigma\left(\sum_{j=1}^{N} A_{2,j}[t]x[j]\right) \\ 0 \end{bmatrix}. \tag{73}$$

As the (d_x+2)th component resets to zero, we use it to compute the third sum $w_3[t]\sigma\left(\sum A_{3,j}[t]x[j]\right)$ and repeat until we obtain the final network $\mathcal{N}$ such that

$$\mathcal{N}(x)[t] = \begin{bmatrix} x[t] \\ \sum_{i=1}^{M} w_i[t]\sigma\left(\sum_{t=1}^{N} A_{i,j}[t]x[j]\right) \\ 0 \end{bmatrix}. \tag{74}$$

■

The following lemma fills the gap between a modified TBRNN and a modified BRNN.

Lemma 21 *Let $\tilde{\mathcal{N}}$ be a modified TBRNN that computes* (69) *and $K \subset \mathbb{R}^{d_x}$ be a compact set. Then for any $\epsilon > 0$ there exists a modified BRNN $\bar{\mathcal{N}}$ of width $d_x + 2 + \gamma(\sigma)$ such that*

$$\sup_{x\in K^N} \left\| \tilde{\mathcal{N}}(x) - \bar{\mathcal{N}}(x) \right\| < \epsilon, \tag{75}$$

where $\gamma(\mathrm{ReLU}) = 0$, $\gamma(\sigma) = 1$ for non-degenerating activation σ.
Moreover, there exists a BRNN $\mathcal{N}$ of width $d_x + 3 + \alpha(\sigma)$ such that

$$\sup_{x\in K^N} \left\| \tilde{\mathcal{N}}(x) - \mathcal{N}(x) \right\| < \epsilon, \tag{76}$$

where $\alpha(\sigma) = \begin{cases} 0 & \sigma \text{ is ReLU or a non-degenerating function with } \sigma(z_0) = 0. \\ 1 & \sigma \text{ is a non-degenerating function with } \sigma(z_0) \neq 0. \end{cases}$

Proof We omit these details because we only need to construct a modified RNN that approximates (67) and (68) using Lemma 13. As only the forward or backward modified RNN cell is used in the proof of Lemma 19, it is enough for the modified BRNN to approximate either the forward or backward modified TRNN. Thus, it follows from Lemma 13. Lemma 3 provides the second part of this theorem. ■

Finally, we obtain the universal approximation theorem of the BRNN from the previous results.

Theorem 22 (Universal approximation theorem of deep BRNN 1) *Let $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ be a continuous sequence to sequence function and σ be a non-degenerate activation function. Then for any $\epsilon > 0$ and compact subset $K \subset \mathbb{R}^{d_x}$, there exists a deep BRNN $\mathcal{N}$ of width $d_x + d_y + 3 + \alpha(\sigma)$, such that*

$$\sup_{x \in K^N} \sup_{1 \le t \le N} \|f(x)[t] - \mathcal{N}(x)[t]\| < \epsilon, \tag{77}$$

where $\alpha(\sigma) = \begin{cases} 0 & \sigma \text{ is ReLU or a non-degenerating function with } \sigma(z_0) = 0. \\ 1 & \sigma \text{ is a non-degenerating function with } \sigma(z_0) \neq 0. \end{cases}$

Proof As in the proof of Theorem 5, we set $d_y = 1$ for notational convenience. According Lemma 20, there exists a modified TBRNN $\tilde{\mathcal{N}}$ of width $d_x + 2$ such that

$$\sup_{x \in K^n} \left\| f(x) - \tilde{\mathcal{N}}(x) \right\| < \frac{\epsilon}{2}. \tag{78}$$

Lemma 21 implies that there exists a BRNN of width $d_x + 4 + \alpha(\sigma)$ such that

$$\sup_{x \in K^n} \left\| \tilde{\mathcal{N}}(x) - \mathcal{N}(x) \right\| < \frac{\epsilon}{2}. \tag{79}$$

The triangle inequality leads to

$$\sup_{x \in K^n} \|f(x) - \mathcal{N}(x)\| < \epsilon. \tag{80}$$

■

In the L^p space, we have similar results with RNNs much more straightforward than in continuous function spaces as the only encoder needs bidirectional flow.

Theorem 23 (Universal approximation theorem of deep BRNN 2) *Let $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}^{d_y \times N}$ be a sequence-to-sequence function in $L^p\left(\mathbb{R}^{d_x \times N}, \mathbb{R}^{d_y \times N}\right)$ for $1 \le p < \infty$, and σ be a non-degenerate activation function with the non-degenerating point z_0. Then, for any $\epsilon > 0$ and compact subset $K \subset \mathbb{R}^{d_x}$, there exists a deep BRNN $\mathcal{N}$ of width $\max\{d_x + 1, d_y\} + 1$ satisfying*

$$\sup_{1 \le t \le N} \|f(x)[t] - \mathcal{N}(x)[t]\|_{L^p(K)} < \epsilon. \tag{81}$$

Moreover, if the activation σ is ReLU, there exists a deep BRNN $\mathcal{N}$ of width $\max\{d_x + 1, d_y\}$ satisfying

$$\sup_{1 \le t \le N} \|f(x)[t] - \mathcal{N}(x)[t]\|_{L^p\left(\mathbb{R}^{d_x}\right)} < \epsilon. \tag{82}$$

Proof Recall that in the RNN case, $x[1:t] \in K^N$ is encoded as a concatenation of a decimal representation of each $x[1], x[2], \ldots, x[t]$, cutting it off at a certain number of digits. The backward process carries the same work, and then the encoded results will be connected. Since the cells constructing the encoders are the same as equation (48), we omit the further step. ■

7. Discussion

We proved the universal approximation theorem and calculated the upper bound of the minimum width of an RNN, an LSTM, a GRU, and a BRNN. In this section, we illustrate how our results support the performance of a recurrent network.

We show that an RNN needs a width of at most $d_x + d_y + 4$ to approximate a function from a sequence of d_x-dimensional vectors to a sequence of d_y-dimensional vectors. The upper bound of the minimum width of the network depends only on the input and output dimensions, regardless of the length of the sequence. The independence of the sequence length indicates that the recurrent structure is much more effective in learning a function on sequential inputs. To approximate a function defined on a long sequence, a network with a feed-forward structure requires a wide width proportional to the length of the sequence. For example, an MLP should have a wider width than Nd_x if it approximates a function $f : \mathbb{R}^{d_x \times N} \to \mathbb{R}$ defined on a sequence (Johnson, 2019). However, with the recurrent structure, it is possible to approximate via a narrow network of width $d_x + 1$ regardless of the length, because the minimum width is independent of the length N. This suggests that the recurrent structure, which transfers information between different time steps in the same layer, is crucial for success with sequential data.

From a practical point of view, this fact further implies that there is no need to limit the length of the time steps that affect dynamics to learn the internal dynamics between sequential data. For instance, suppose that a pair of long sequential data $(x[t])$ and $(y[t])$ have an unknown relation $y[t] = f(x[t-p], x[t-p+1], \ldots, x[t])$. Even without prior knowledge of f and p, a deep RNN learns the relation if we train the network with inputs $x[1:t]$ and outputs $y[t]$. The MLP cannot reproduce the result because the required width increases proportionally to p, which is an unknown factor. The difference between these networks theoretically supports that recurrent networks are appropriate when dealing with sequential data whose underlying dynamics are unknown in the real world.

8. Conclusion

In this study, we investigated the universality and upper bound of the minimum width of deep RNNs. The upper bound of the minimum width serves as a theoretical basis for the effectiveness of deep RNNs, especially when underlying dynamics of the data are unknown.

Our methodology enables various follow-up studies, as it connects an MLP and a deep RNN. For example, the framework disentangles the time dependency of output sequence of an RNN. This makes it feasible to investigate a trade-off between width and depth in the representation ability or error bounds of the deep RNN, which has not been studied because of the entangled flow with time and depth. In addition, we separated the required width into three parts: one maintains inputs and results, another resolves the time dependency, and the third modifies the activation. Assuming some underlying dynamics in the output sequence, such as an open dynamical system, we expect to reduce the required minimum width on each part because there is a natural dependency between the outputs, and the inputs are embedded in a specific way by the dynamics.

However, as LSTMs and GRUs have multiple hidden states in the cell process, they may have a smaller minimum width than the RNN. By constructing an LSTM and a GRU

to use the hidden states to save data and resolve the time dependency, we hope that our techniques demonstrated in the proof help analyze why these networks have a better result in practice and suffer less from long-term dependency.

Acknowledgement

This work was supported by the Challengeable Future Defense Technology Research and Development Program through ADD[No. 915020201], the NRF grant[2012R1A2C3010887] and the MSIT/IITP[No. 2021-0-01343, Artificial Intelligence Graduate School Program(SNU)].

Symbol	Description
σ	Activation function
σ_I	Modified activation function
x	Input sequence of vectors
y	Output sequence of vectors
t	Order of element in a sequence
$a[t]_i$	i-th component of t-element of a sequence a
d_x	Dimension of input vector
d_s	Dimension of hidden state in RNN, or width
d_y	Dimension of output vector
N	Length of the input and output sequence
K	A compact subset of $\mathbb{R}^{d_x}$
$O_{m,n}$	Zero matrix of size $m \times n$
$\mathbf{0}_k$	Zero vector of size k, $\mathbf{0}_k = \underbrace{\begin{bmatrix}0 & 0 \cdots & 0\end{bmatrix}}_{k}^{T}$
$\mathbf{1}_k$	One vector of size k, $\mathbf{1}_k = \underbrace{\begin{bmatrix}1 & 1 \cdots & 1\end{bmatrix}}_{k}^{T}$
I_k	Identity matrix of size $k \times k$
RNN	Recurrent Neural Network defined in page 6
TRNN	Time-enhanced Recurrent Neural Network, defined by replacing recurrent cell with time-enhanced recurrent cell in RNN. Time-enhenced cell and TRN are defined in page 13
BRNN	Bidirectional Recurrent Neural Network defined in page 7
TBRNN	Time-enhanced Bidirectional Recurrent Neural Network defined in page 13

Appendix A. Notations

Appendix B. Proof of the Lemma 3

Without loss of generality, we may assume $\bar{\mathcal{P}}$ is an identity map and $I = \{1, 2, \ldots, k\}$. Let $\bar{\mathcal{R}}(x)[t+1] = \sigma_I\left(\bar{A}\bar{\mathcal{R}}(x)[t] + \bar{B}x[t+1] + \bar{\theta}\right)$ be a given modified RNN cell, and $\mathcal{Q}(x)[t] = \bar{Q}x[t]$ be a given token-wise linear projection map. We use notations $O_{m,n}$ and $\mathbf{1}_m$ to denote zero matrix in $\mathbb{R}^{m\times n}$ and one vector in $\mathbb{R}^m$ respectively. Sometimes we omit $O_{m,n}$ symbol in some block-diagonal matrices if the size of the zero matrix is clear.

Case 1: $\sigma(z_0) = 0$
Let $\mathcal{P}$ be the identity map. For $\delta > 0$ define $\mathcal{R}_1^\delta$ as

$$\mathcal{R}_1^\delta\left(x[t+1]\right) \coloneqq \sigma\left(\delta\bar{B}x[t+1] + \delta\bar{\theta} + z_0\mathbf{1}_d\right). \tag{83}$$

Since σ is non-degenerating at z_0 and σ' is continuous at z_0, we have

$$\mathcal{R}_1^\delta \circ \mathcal{P}(x)[t+1] = \delta\sigma'(z_0)\left(\bar{B}x[t+1] + \bar{\theta}\right) + o(\delta). \tag{84}$$

Then construct a second cell to compute transition as

$$\mathcal{R}_2^\delta(x)[t+1] = \sigma\left(\tilde{A}\mathcal{R}_2^\delta(x)[t] + \frac{1}{\sigma'(z_0)}\begin{bmatrix}\delta^{-1}I_k & \\ & I_{d-k}\end{bmatrix}x[t+1] + \begin{bmatrix}\mathbf{0}_k \\ z_0\mathbf{1}_{d-k}\end{bmatrix}\right), \tag{85}$$

where $\tilde{A} = \begin{bmatrix}I_k & \\ & \delta I_{d-k}\end{bmatrix}\bar{A}\begin{bmatrix}I_k & \\ & \frac{1}{\delta\sigma'(z_0)}I_{d-k}\end{bmatrix}$.
After that, the first output of $\mathcal{R}_2^\delta\mathcal{R}_1^\delta\mathcal{P}(x)$ becomes

$$\begin{aligned}
\mathcal{R}_2^\delta\mathcal{R}_1^\delta\mathcal{P}(x)[1] &= \sigma\left(\frac{1}{\sigma'(z_0)}\begin{bmatrix}\delta^{-1}I_k & \\ & I_{d-k}\end{bmatrix}\mathcal{R}_1^\delta(x)[1] + \begin{bmatrix}\mathbf{0}_k \\ z_0\mathbf{1}_{d-k}\end{bmatrix}\right) && (86)\\
&= \sigma\left(\begin{bmatrix}\left(\bar{B}x[1]+\bar{\theta}\right)_{1:k} + \delta^{-1}o(\delta) \\ \left(z_0\mathbf{1}_{d-k} + \delta(\bar{B}x[1]+\bar{\theta})\right)_{k+1:d} + o(\delta)\end{bmatrix}\right) && (87)\\
&= \begin{bmatrix}\sigma\left(\bar{B}x[1]+\bar{\theta}\right)_{1:k} + o(1) \\ \sigma'(z_0)\delta\left(\bar{B}x[1]+\bar{\theta}\right)_{k+1:d} + o(\delta)\end{bmatrix} && (88)\\
&= \begin{bmatrix}\bar{\mathcal{R}}(x)[1]_{1:k} + o(1) \\ \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[1]_{k+1:d} + o(\delta)\end{bmatrix}. && (89)
\end{aligned}$$

Now use mathematical induction on time t to compute $\mathcal{R}_2^\delta\mathcal{R}_1^\delta\mathcal{P}(x)$ assuming

$$\mathcal{R}_2^\delta\mathcal{R}_1^\delta\mathcal{P}(x)[t] = \begin{bmatrix}\bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[t]_{k+1:d} + o(\delta)\end{bmatrix}. \tag{90}$$

From a direct calculation, we attain

$$\begin{aligned}
&\frac{1}{\sigma'(z_0)}\begin{bmatrix}\delta^{-1}I_k & \\ & I_{d-k}\end{bmatrix}\mathcal{R}_1^\delta\mathcal{P}(x)[t+1] + \begin{bmatrix}\mathbf{0}_k \\ z_0\mathbf{1}_{d-k}\end{bmatrix} && (91)\\
&= \frac{1}{\sigma'(z_0)}\begin{bmatrix}\delta^{-1}I_k & \\ & I_{d-k}\end{bmatrix}\left(\delta\sigma'(z_0)\left(\bar{B}x[t+1]+\bar{\theta}\right) + o(\delta)\right) + \begin{bmatrix}\mathbf{0}_k \\ z_0\mathbf{1}_{d-k}\end{bmatrix} && (92)\\
&= \begin{bmatrix}\bar{B}x[t+1]_{1:k} + \bar{\theta}_{1:k} + \delta^{-1}o(\delta) \\ z_0\mathbf{1}_{d-k} + \delta\left(\bar{B}x[t+1]+\bar{\theta}\right)_{k+1:d} + o(\delta)\end{bmatrix}, && (93)
\end{aligned}$$

and

$$\begin{aligned}
&\tilde{A}\mathcal{R}_2^\delta\mathcal{R}_1^\delta\mathcal{P}(x)[t] && (94)\\
&= \begin{bmatrix}I_k & \\ & \delta I_{d-k}\end{bmatrix}\bar{A}\begin{bmatrix}I_k & \\ & \frac{1}{\delta\sigma'(z_0)}I_{d-k}\end{bmatrix}\begin{bmatrix}\bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[t]_{k+1:d} + o(\delta)\end{bmatrix} && (95)\\
&= \begin{bmatrix}I_k & \\ & \delta I_{d-k}\end{bmatrix}\bar{A}\begin{bmatrix}\bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \bar{\mathcal{R}}(x)[t]_{k+1:d} + o(1)\end{bmatrix} && (96)\\
&= \begin{bmatrix}\left(\bar{A}\bar{\mathcal{R}}(x)[t]\right)_{1:k} + o(1) \\ \delta\left(\bar{A}\bar{\mathcal{R}}(x)[t]\right)_{k+1:d} + o(\delta)\end{bmatrix}. && (97)
\end{aligned}$$

With the sum of above two results, we obtain the induction hypothesis (90) for $t+1$,

$$\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t+1] \tag{98}$$

$$= \sigma\left(\tilde{A}\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t] + \frac{1}{\sigma'(z_0)}\begin{bmatrix}\delta^{-1} I_k & \\ & I_{d-k}\end{bmatrix} \mathcal{R}_1^\delta \mathcal{P}(x)[t+1] + \begin{bmatrix}\mathbf{0}_k \\ z_0 \mathbf{1}_{d-k}\end{bmatrix}\right) \tag{99}$$

$$= \sigma\begin{bmatrix}\left(\bar{A}\bar{\mathcal{R}}(x)[t]\right)_{1:k} + \bar{B}x[t+1]_{1:k} + \bar{\theta}_{1:k} + o(1) \\ z_0 \mathbf{1}_{d-k} + \delta\left(\bar{A}\bar{\mathcal{R}}(x)[t]\right)_{k+1:d} + \delta\left(\bar{B}x[t+1] + \bar{\theta}\right)_{k+1:d} + o(\delta)\end{bmatrix} \tag{100}$$

$$= \begin{bmatrix}\bar{\mathcal{R}}(x)[t+1]_{1:k} + o(1) \\ \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[t+1]_{k+1:d} + o(\delta)\end{bmatrix}. \tag{101}$$

Setting $\mathcal{Q}^\delta = \bar{Q}\begin{bmatrix} I_k & \\ & \frac{1}{\sigma'(z_0)\delta} I_{d-k}\end{bmatrix}$ and choosing δ small enough complete the proof:

$$\mathcal{Q}^\delta \mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t] = \bar{Q}\begin{bmatrix}\bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \bar{\mathcal{R}}(x)[t]_{k+1:d} + o(1)\end{bmatrix} = \bar{\mathcal{Q}}\bar{\mathcal{R}}(x)[t] + o(1) \to \bar{\mathcal{Q}}\bar{\mathcal{R}}(x)[t]. \tag{102}$$

Case 2: $\sigma(z_0) \neq 0$
When $\sigma(z_0) \neq 0$, there is $\sigma(z_0)$ term independent of δ in the Taylor expansion of $\sigma(z_0+\delta x) = \sigma(z_0) + \delta\sigma'(z_0)x + o(\delta)$. An additional width removes the term in this case; hence we need a lifting map $\mathcal{P} : \mathbb{R}^{d\times N} \to \mathbb{R}^{(d+1)\times N}$:

$$\mathcal{P}(x)[t] = \begin{bmatrix} x[t] \\ 0\end{bmatrix}. \tag{103}$$

Now for $\delta > 0$ define $\mathcal{R}_1^\delta$ as

$$\mathcal{R}_1^\delta(X) \coloneqq \sigma\left(\delta\begin{bmatrix}\bar{B} & \\ & 0\end{bmatrix} X + \delta\begin{bmatrix}\bar{\theta} \\ 0\end{bmatrix} + z_0 \mathbf{1}_{d+1}\right). \tag{104}$$

As in the previous case, we have

$$\mathcal{R}_1^\delta \circ \mathcal{P}(x)[t+1] = \begin{bmatrix}\sigma(z_0)\mathbf{1}_d + \delta\sigma'(z_0)\left(\bar{B}x[t+1] + \bar{\theta}\right) + o(\delta) \\ \sigma(z_0)\end{bmatrix}, \tag{105}$$

and construct a second cell $\mathcal{R}_2^\delta$ to compute

$$\mathcal{R}_2^\delta(x)[t+1] = \sigma\left(\tilde{A}\mathcal{R}_2^\delta(x)[t] + \frac{1}{\sigma'(z_0)}\begin{bmatrix}\delta^{-1} I_k & \\ & I_{d+1-k}\end{bmatrix} x[t+1] + \begin{bmatrix}-\frac{\sigma(z_0)}{\delta\sigma'(z_0)}\mathbf{1}_k \\ z_0\mathbf{1}_{d+1-k} - \frac{\sigma(z_0)}{\sigma'(z_0)}\mathbf{1}_{d+1-k}\end{bmatrix}\right), \tag{106}$$

where $\tilde{A} = \begin{bmatrix} I_k & \\ & \delta I_{d+1-k} \end{bmatrix} \begin{bmatrix} \bar{A} & \\ & 0 \end{bmatrix} \begin{bmatrix} I_k & & \\ & \frac{1}{\delta\sigma'(z_0)} I_{d-k} & -\frac{1}{\delta\sigma'(z_0)} \mathbf{1}_{d-k} \\ & & 0 \end{bmatrix}$.

After that, the first output of $\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t]$ becomes

$$\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[1] = \sigma\left(\frac{1}{\sigma'(z_0)} \begin{bmatrix} \delta^{-1} I_k & \\ & I_{d+1-k} \end{bmatrix} \mathcal{R}_1^\delta(x)[1] + \begin{bmatrix} -\frac{\sigma(z_0)}{\delta\sigma'(z_0)} \mathbf{1}_k \\ z_0 \mathbf{1}_{d+1-k} - \frac{\sigma(z_0)}{\sigma'(z_0)} \mathbf{1}_{d+1-k} \end{bmatrix} \right) \tag{107}$$

$$= \sigma\left(\begin{bmatrix} \left(\bar{B}x[1] + \bar{\theta}\right)_{1:k} + o(\delta) \\ \left(z_0 \mathbf{1}_{d-k} + \delta(\bar{B}x[1] + \bar{\theta}\right)_{k+1:d} + o(\delta) \\ z_0 \end{bmatrix} \right) \tag{108}$$

$$= \begin{bmatrix} \sigma\left(\bar{B}x[1] + \bar{\theta}\right)_{1:k} + o(1) \\ \sigma(z_0)\mathbf{1}_{d-k} + \sigma'(z_0)\delta\left(\bar{B}x[1] + \bar{\theta}\right)_{k+1:d} + o(\delta) \\ \sigma(z_0) \end{bmatrix} \tag{109}$$

$$= \begin{bmatrix} \bar{\mathcal{R}}(x)[1]_{1:k} + o(1) \\ \sigma(z_0)\mathbf{1}_{d-k} + \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[1]_{k+1:d} + o(\delta) \\ \sigma(z_0) \end{bmatrix}. \tag{110}$$

Assume $\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)$ and use mathematical induction on time t.

$$\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t] = \begin{bmatrix} \bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \sigma(z_0)\mathbf{1}_{d-k} + \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[t]_{k+1:d} + o(\delta) \\ \sigma(z_0) \end{bmatrix}. \tag{111}$$

Direct calculation yields

$$\frac{1}{\sigma'(z_0)} \begin{bmatrix} \delta^{-1} I_k & \\ & I_{d+1-k} \end{bmatrix} \mathcal{R}_1^\delta \mathcal{P}(x)[t+1] + \begin{bmatrix} -\frac{\sigma(z_0)}{\delta\sigma'(z_0)} \mathbf{1}_k \\ z_0 \mathbf{1}_{d+1-k} - \frac{\sigma(z_0)}{\sigma'(z_0)} \mathbf{1}_{d+1-k} \end{bmatrix} \tag{112}$$

$$= \begin{bmatrix} \bar{B}x[t+1]_{1:k} + \bar{\theta}_{1:k} + o(1) \\ z_0 \mathbf{1}_{d-k} + \delta\left(\bar{B}x[t+1] + \bar{\theta}\right)_{k+1:d} + o(\delta) \\ z_0 \end{bmatrix}, \tag{113}$$

and

$$\tilde{A}\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t] \tag{114}$$

$$= \tilde{A} \begin{bmatrix} \bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \sigma(z_0)\mathbf{1}_{d-k} + \sigma'(z_0)\delta\bar{\mathcal{R}}(x)[t]_{k+1:d} + o(\delta) \\ \sigma(z_0) \end{bmatrix} \tag{115}$$

$$= \begin{bmatrix} I_k & \\ & \delta I_{d+1-k} \end{bmatrix} \begin{bmatrix} \bar{A} & \\ & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \bar{\mathcal{R}}(x)[t]_{k+1:d} + o(1) \\ 0 \end{bmatrix} \tag{116}$$

$$= \begin{bmatrix} \left(\bar{A}\bar{\mathcal{R}}(x)[t]\right)_{1:k} + o(1) \\ \delta\left(\bar{A}\bar{\mathcal{R}}(x)[t]\right)_{k+1:d} + o(\delta) \\ 0 \end{bmatrix}. \tag{117}$$

Adding two terms in (106), we obtain the induction hypothesis (111) for $t+1$,

$$\mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t+1] = \begin{bmatrix} \bar{\mathcal{R}}(x)[t+1]_{1:k} + o(1) \\ \sigma(z_0)\mathbf{1}_{d-k} + \sigma'(z_0)\delta \bar{\mathcal{R}}(x)[t+1]_{k+1:d} + o(\delta) \\ \sigma(z_0) \end{bmatrix}. \tag{118}$$

Setting $\mathcal{Q}^\delta = \begin{bmatrix} \bar{Q} & 0 \end{bmatrix} \begin{bmatrix} I_k & & \\ & \frac{1}{\sigma'(z_0)\delta} I_{d-k} & -\frac{1}{\sigma'(z_0)\delta}\mathbf{1}_{d-k} \\ & & 0 \end{bmatrix}$ and choosing δ small enough complete the proof:

$$\mathcal{Q}^\delta \mathcal{R}_2^\delta \mathcal{R}_1^\delta \mathcal{P}(x)[t] = \begin{bmatrix} \bar{Q} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathcal{R}}(x)[t]_{1:k} + o(1) \\ \bar{\mathcal{R}}(x)[t]_{k+1:d} + o(1) \\ 0 \end{bmatrix} = \bar{\mathcal{Q}}\bar{\mathcal{R}}(x)[t] + o(1) \to \bar{\mathcal{Q}}\bar{\mathcal{R}}(x)[t]. \tag{119}$$

Appendix C. Proof of the Lemma 6

It suffices to show that there exists a modified RNN $\mathcal{N}$ that computes

$$\mathcal{N}(x)[N] = \begin{bmatrix} x[N] \\ \sum_{t=1}^N A[t]x[t] \end{bmatrix}, \tag{120}$$

for given matrices $A[1], \ldots, A[N] \in \mathbb{R}^{1\times d_x}$.

RNN should have multiple layers to implement the arbitrary linear combination. To overcome the complex time dependency deriving from deep structures and explicitly formulate the results of deep modified RNN, we force A and B to use the information of the previous time step in a limited way. Define the modified RNN cell at l-th layer $\mathcal{R}_l$ as

$$\mathcal{R}_l(x)[t+1] = A_l \mathcal{R}_l(x)[t] + B_l x[t+1] \tag{121}$$

where $A_l = \begin{bmatrix} O_{d_x,d_x} & O_{d_x,1} \\ O_{1,d_x} & 1 \end{bmatrix}$, $B_l = \begin{bmatrix} I_{d_x} & O_{d_x,1} \\ b_l & 1 \end{bmatrix}$ for $b_l \in \mathbb{R}^{1\times d_x}$.

Construct a modified RNN $\mathcal{N}_L$ for each $L \in \mathbb{N}$ as

$$\mathcal{N}_L \coloneqq \mathcal{R}_L \circ \mathcal{R}_{L-1} \circ \cdots \circ \mathcal{R}_1, \tag{122}$$

and denote the output of $\mathcal{N}_L$ at each time m for an input sequence $x' = \begin{bmatrix} x \\ 0 \end{bmatrix} \in \mathbb{R}^{d_x+1}$ of embedding of x:

$$T(n,m) \coloneqq \mathcal{N}_n\left(x'\right)[m]. \tag{123}$$

Then we have the following lemma.

Lemma 24 *Let $T(n,m)$ be the matrix defined by (123). Then we have*

$$T(n,m) = \begin{bmatrix} x[m] \\ \sum_{i=1}^\infty \sum_{j=1}^\infty \binom{n+m-i-j}{n-i} b_i x[j] \end{bmatrix}, \tag{124}$$

where $\binom{n}{k}$ means binomial coefficient $\frac{n!}{k!(n-k)!}$ for $n \geq k$. We define $\binom{n}{k} = 0$ for the case of $k > n$ or $n < 0$ for notational convenience.

Proof Since there is no activation in modified RNN (121), $T(n,m)$ has the form of

$$T(n,m) = \begin{bmatrix} x_m \\ \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_{i,j}^{n,m} b_i x[j] \end{bmatrix}. \tag{125}$$

From the definition of the modified RNN cell and T, we first show that α satisfies the recurrence relation

$$\alpha_{i,j}^{n,m} = \begin{cases} \alpha_{i,j}^{n-1,m} + \alpha_{i,j}^{n,m-1} + 1, & \text{if } n = i \text{ and } m = j, \\ \alpha_{i,j}^{n-1,m} + \alpha_{i,j}^{n,m-1}, & \text{otherwise} \end{cases} \tag{126}$$

using mathematical induction on n, m in turn. Initially, $T(0,m) = \begin{bmatrix} x_m \\ 0 \end{bmatrix}$, $T(n,0) = \begin{bmatrix} O_{d_x,1} \\ 0 \end{bmatrix}$ by definition, and (125) holds when $n = 0$. Now assume (125) holds for $n \le N$, any m. To show that (125) holds for $n = N+1$ and any m, use mathematical induction on m. By definition, we have $\alpha_{i,j}^{n,0} = 0$ for any n. Thus (125) holds when $n = N+1$ and $m = 0$. Assume it holds for $n = N+1$ and $m \le M$. Then

$$\begin{aligned} &T(N+1, M+1) \\ &= \begin{bmatrix} O_{d_x,d_x} & O_{d_x,1} \\ O_{1,d_x} & 1 \end{bmatrix} \begin{bmatrix} x_M \\ \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_{i,j}^{N+1,M} x_j \end{bmatrix} + \begin{bmatrix} I_{d_x} & O_{d_x,1} \\ b_{N+1} & 1 \end{bmatrix} \begin{bmatrix} x_{M+1} \\ \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_{i,j}^{N,M+1} x_j \end{bmatrix} \\ &= \begin{bmatrix} O_{d_x,1} \\ \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_{i,j}^{N+1,M} b_i x_j \end{bmatrix} + \begin{bmatrix} x_{M+1} \\ b_{N+1} x_{M+1} + \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \left(\alpha_{i,j}^{N+1,M} + \alpha_{i,j}^{N,M+1}\right) b_i x_j \end{bmatrix}. \end{aligned} \tag{127}$$

Hence the relation holds for $n = N+1$ and any $m > 0$.

Now remains to show

$$\alpha_{i,j}^{n,m} = \begin{cases} \binom{m+n-i-j}{n-i} & \text{if } 1 \le i \le n,\ 1 \le j \le m \\ 0 & \text{otherwise} \end{cases} \tag{128}$$

From the initial condition of α, we know $\alpha_{i,j}^{0,m} = \alpha_{i,j}^{n,0} = 0$ for all $n, m \in \mathbb{N}$. After some direct calculation with the recurrence relation (125) of α, we have

i) If $n < i$ or $m < j$, $\alpha_{i,j}^{n,m} = 0$ as $\alpha_{i,j}^{n,m} = \alpha_{i,j}^{n-1,m} + \alpha_{i,j}^{n,m-1}$.

ii) $\alpha_{i,j}^{i,j} = \alpha_{i,j}^{i-1,j} + \alpha_{i,j}^{i,j-1} + 1 = 1$.

iii) $\alpha_{i,j}^{i,m} = \alpha_{i,j}^{i-1,m} + \alpha_{i,j}^{i,m-1} = \alpha_{i,j}^{i,m-1}$ implies $\alpha_{i,j}^{i,m} = 1$ for $m > j$.

iv) Similarly, $\alpha_{i,j}^{n,j} = \alpha_{i,j}^{n-1,j} + \alpha_{i,j}^{n,j-1} = \alpha_{i,j}^{n-1,j}$ implies $\alpha_{i,j}^{n,j} = 1$ for $n > i$.

Now use mathematical induction on $n + m$ starting from $n + m = i + j$ to show $\alpha_{i,j}^{n,m} = \binom{m+n-i-j}{n-i}$ for $n \ge i$, $m \ge j$.

i) $n + m = i + j$ holds only if $n = i$, $m = j$ for $n \ge i$, $m \ge j$. In the case, $\alpha_{i,j}^{i,j} = 1 = \binom{m+n-i-j}{n-i}$.

ii) Assume that (128) holds for any n, m with $n+m=k$ as induction hypothesis. Now suppose $n+m=k+1$ for given n, m. If $n=i$ or $m=j$ we already know $\alpha_{i,j}^{n,m}=1=\binom{m+n-i-j}{n-i}$. Otherwise $n-1\geq i$, $m-1\geq j$, and we have

$$\begin{aligned}\alpha_{i,j}^{n,m} &= \alpha_{i,j}^{n-1,m}+\alpha_{i,j}^{n,m-1}\\ &= \binom{m+n-1-i-j}{n-1-i}+\binom{m+n-1-i-j}{n-i}\\ &= \binom{m+n-i-j}{n-i},\end{aligned} \tag{129}$$

which completes the proof.

■

We have computed the output of modified RNN $\mathcal{N}_N$ such that

$$\mathcal{N}_N\left(x'\right)[N]=\begin{bmatrix} x[N] \\ \sum_{i=1}^{N}\sum_{j=1}^{N}\binom{2n-i-j}{n-i}b_i x[j]\end{bmatrix}. \tag{130}$$

If the square matrix $\Lambda_N=\left\{\binom{2n-i-j}{n-i}\right\}_{1\leq i,j\leq N}$ has inverse $\Lambda_N^{-1}=\{\lambda_{i,j}\}_{1\leq i,j\leq N}$, $b_i=\sum_{t=1}^{N}\lambda_{t,i}A[t]$ satisfies

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=1}^{N}\binom{2n-i-j}{n-i}b_i x[j] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n}\binom{2n-i-j}{n-i}\lambda_{t,i}A[t]x[j]\\ &= \sum_{j=1}^{n}\sum_{t=1}^{n}\left[\sum_{i=1}^{n}\binom{2n-i-j}{n-i}\lambda_{t,i}\right]A[t]x[j]\\ &= \sum_{j=1}^{n}\sum_{t=1}^{n}\delta_{j,t}A[t]x[j]\\ &= \sum_{j=1}^{n}A[j]x[j],\end{aligned}$$

where δ is the Kronecker delta function.

The following lemma completes the proof.

Lemma 25 *Matrix* $\Lambda_n=\left\{\binom{2n-i-j}{n-i}\right\}_{1\leq i,j\leq n}\in\mathbb{R}^{n\times n}$ *is invertible.*

Proof Use mathematical induction on n. Λ_1 is a trivial case. Assume Λ_n is invertible.

$$\Lambda_{n+1}=\begin{bmatrix}\binom{2n}{n} & \binom{2n-1}{n} & \binom{2n-2}{n} & \cdots & \binom{n+1}{n} & \binom{n}{n}\\ \binom{2n-1}{n-1} & \binom{2n-2}{n-1} & \binom{2n-3}{n-1} & \cdots & \binom{n}{n-1} & \binom{n-1}{n-1}\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ \binom{n+1}{1} & \binom{n}{1} & \binom{n-1}{1} & \cdots & \binom{2}{1} & \binom{1}{1}\\ \binom{n}{0} & \binom{n-1}{0} & \binom{n-2}{0} & \cdots & \binom{1}{0} & \binom{0}{0}\end{bmatrix}. \tag{131}$$

Applying elementary row operation to Λ_{n+1} by multiplying the matrix E on the left and elementary column operation to $E\Lambda_{n+1}$ by multiplying the matrix E^T on the right where

$$E = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -1 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}, \tag{132}$$

we obtain the following relation:

$$E\Lambda_{n+1}E^T = \begin{bmatrix} \binom{2n-2}{n-1} & \binom{2n-3}{n-1} & \binom{2n-4}{n-1} & \dots & \binom{n-1}{n-1} & 0 \\ \binom{2n-3}{n-2} & \binom{2n-4}{n-2} & \binom{2n-5}{n-2} & \dots & \binom{n-2}{n-2} & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \binom{n-1}{0} & \binom{n-2}{0} & \binom{n-3}{0} & \dots & \binom{0}{0} & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} = \begin{bmatrix} \Lambda_n & O_{n,1} \\ O_{1,n} & 1 \end{bmatrix}. \tag{133}$$

Hence Λ_{n+1} is invertible by the induction hypothesis. ■

Corollary 26 *The following matrix $\Lambda_{n,k} \in \mathbb{R}^{k\times n}$ is full-rank.*

$$\Lambda_{n,k} = \left\{ \binom{2n-i-j}{n-i} \right\}_{n-k+1\le i\le n, 1\le j\le n}. \tag{134}$$

We will use the matrix $\Lambda_{n,k}$ in the proof of Lemma 11 to approximate a sequence-to-sequence function.

Appendix D. Proof of Lemma 11

Define token-wise lifting map $\mathcal{P} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x+1}$ and modified TRNN cell $\mathcal{TR}_l : \mathbb{R}^{(d_x+1)\times N} \to \mathbb{R}^{(d_x+1)\times N}$ as in the proof of Lemma 6:

$$\mathcal{P}(x)[t] = \begin{bmatrix} x[m] \\ 0 \end{bmatrix}, \tag{135}$$

$$\mathcal{TR}_l(X)[t+1] = A_l\mathcal{TR}_l(X)[t] + B_l[t](X)[t+1], \tag{136}$$

where $A_l = \begin{bmatrix} O_{d_x,d_x} & O_{d_x,1} \\ O_{1,d_x} & 1 \end{bmatrix}$, $B_l[t] = \begin{bmatrix} I_{d_x} & O_{d_x,1} \\ b_l[t] & 1 \end{bmatrix}$ for $b_l[t] \in \mathbb{R}^{1\times d_x}$. Then we have

$$\begin{aligned} T(n,m) &\coloneqq \mathcal{N}_n(x)[m] \\ &= \begin{bmatrix} x[m] \\ \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\binom{n+m-i-j}{n-i} b_i[j]x[j] \end{bmatrix}, \end{aligned} \tag{137}$$

where $x \in \mathbb{R}^{d_x\times N}$ and $\mathcal{N}_L = \mathcal{TR}_L \circ \mathcal{TR}_{L-1} \circ \cdots \circ \mathcal{TR}_1 \circ \mathcal{P}$.

Since for each t, the matrix

$$\Lambda_{N,N-t+1} = \left\{ \binom{2N-i-j}{N-i} \right\}_{t \le i \le N, 1 \le j \le N} = \left\{ \binom{2N-t+1-i-j}{N-j} \right\}_{1 \le i \le N-t+1, 1 \le j \le N} \tag{138}$$

is full-rank, there exist $b_1[t], b_2[t], \ldots, b_N[t]$ satisfying

$$\Lambda_{N,N-t+1} \begin{bmatrix} b_1[t] \\ \vdots \\ b_N[t] \end{bmatrix} = \begin{bmatrix} A_t[N] \\ \vdots \\ A_t[t] \end{bmatrix}, \tag{139}$$

or

$$\sum_{j=1}^{N} \binom{N+k-j-t}{N-j} b_j[t] = A_t[k], \tag{140}$$

for each $k = 1, 2, \ldots, N$. Then we obtain

$$\begin{aligned} T(N,t) &= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \binom{N+t-i-j}{N-i} b_i[j] x[j] \\ &= \sum_{j=1}^{t} \sum_{i=1}^{N} \binom{N+t-i-j}{N-i} b_i[j] x[j] \\ &= \sum_{j=1}^{t} A_j[t] x[j]. \end{aligned} \tag{141}$$

Appendix E. Proof of Lemma 13

As one of the modified TRNNs that computes (31), we use the modified TRNN defined in Appendix D. Specifically, we show that for a given l, there exists a modified RNN of width $d_x + 2 + \gamma(\sigma)$ that approximates the modified TRNN cell $\mathcal{TR}_l : \mathbb{R}^{(d_x+1)\times N} \to \mathbb{R}^{(d_x+1)\times N}$ defined by (135). Suppose $K \subset \mathbb{R}^{d_x}$, $K' \subset \mathbb{R}$ are compact sets and $X \in (K \times K')^N \subset \mathbb{R}^{(d_x+1)\times N}$. Then the output of the TRNN cell $\mathcal{TR}_l$ is

$$\mathcal{TR}_l(X)[t] = \begin{bmatrix} X[t]_{1:d_x} \\ \sum_{j=1}^{t} b_l[j] X[j]_{1:d_x} + \sum_{j=1}^{t} X[j]_{d_x+1} \end{bmatrix}. \tag{142}$$

Without loss of generality, assume $K \subset \left[0, \frac{1}{2}\right]^{d_x}$ and let $\gamma = \gamma(\sigma)$. Let $\mathcal{P} : \mathbb{R}^{d_x+1} \to \mathbb{R}^{d_x+2+\gamma}$ be a token-wise linear map defined by $\mathcal{P}(X) = \begin{bmatrix} X_{1:d_x} \\ 0 \\ X_{d_x+1} \\ \mathbf{0}_\gamma \end{bmatrix}$. Construct the modified recurrent

cells $\mathcal{R}_1, \mathcal{R}_2 : \mathbb{R}^{(d_x+2+\gamma(\sigma))\times N} \to \mathbb{R}^{(d_x+2+\gamma(\sigma))\times N}$ as for $X' \in \mathbb{R}^{(d_x+2+\gamma)\times N}$,

$$\mathcal{R}_1(X')[t+1] = \begin{bmatrix} O_{d_x,d_x} & & \\ & 1 & \\ & & O_{1+\gamma,1+\gamma} \end{bmatrix} \mathcal{R}_1(X')[t] + X'[t+1] + \begin{bmatrix} \mathbf{0}_{d_x} \\ 1 \\ \mathbf{0}_{1+\gamma} \end{bmatrix}, \tag{143}$$

$$\mathcal{R}_2(X')[t+1] = \begin{bmatrix} I_{d_x} & \mathbf{1}_{d_x} & & \\ & 1 & & \\ & & 1 & \\ & & & O_{\gamma,\gamma} \end{bmatrix} X'[t]. \tag{144}$$

Then, by definition for $X \in (K \times K')^N$,

$$\mathcal{R}_2\mathcal{R}_1\mathcal{P}(X)[t] = \begin{bmatrix} X[t]_{1:d_x} + t\mathbf{1}_{d_x} \\ t \\ X[t]_{d_x+1} \\ \mathbf{0}_\gamma \end{bmatrix}. \tag{145}$$

Note that $D_i = \{\mathcal{R}_2\mathcal{R}_1\mathcal{P}(X)[i]_{1:d_x} \mid X \in (K \times K')^N\} = \{X[i]_{1:d_x} + t\mathbf{1}_{d_x} \mid X \in (K \times K')^N\}$ are disjoint each other, $D_i \cap D_j = \phi$ for all $i \neq j$.

By the universal approximation theorem of deep MLP from Hanin and Sellke (2017); Kidger and Lyons (2020), for any $\delta_l > 0$, there exists an MLP $\mathcal{N}_{l,MLP} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x+1}$ of width $d_x + 1 + \gamma$ such that for $v \in \mathbb{R}^{d_x}$,

$$\mathcal{N}_{l,MLP}(v)_{1:d_x} = v \tag{146}$$

$$\sup_{t=1,\dots,N} \sup_{v \in D_t} \|b_l[t]\,(v - t\mathbf{1}d_x) - \mathcal{N}_{l,MLP}(v)_{d_x+1}\| < \delta_l. \tag{147}$$

Since token-wise MLP is implemented by RNN with the same width, there exists an RNN $\mathcal{N}_l : \mathbb{R}^{d_x+2+\gamma} \to \mathbb{R}^{d_x+2+\gamma}$ of width $d_x+2+\gamma$ whose components all but (d_x+2)-th construct $\mathcal{N}_{l,MLP}$ so that for all $X' \in \mathbb{R}^{d_x+2+\gamma}$,

$$\mathcal{N}_l(X')[t] = \begin{bmatrix} \mathcal{N}_{l,MLP}\,(X'[t]_{1:d_x}) \\ X'[t]_{d_x+2} \\ \mathbf{0}_\gamma \end{bmatrix}. \tag{148}$$

Then for $X \in (K \times K')^N$ we have

$$\mathcal{N}_l\mathcal{R}_2\mathcal{R}_1\mathcal{P}(X)[t] = \begin{bmatrix} \mathcal{N}_{l,MLP}\,(X[t]_{1:d_x} + t\mathbf{1}_{d_x}) \\ X[t]_{d_x+1} \\ \mathbf{0}_\gamma \end{bmatrix}. \tag{149}$$

Finally, define a recurrent cell $\mathcal{R}_3 : \mathbb{R}^{d_x+2+\gamma} \to \mathbb{R}^{d_x+2+\gamma}$ of width $d_x + 2 + \gamma$ as

$$\mathcal{R}_3(X')[t+1] = \begin{bmatrix} O_{d_x+1,d_x+1} & & \\ & 1 & \\ & & O_{\gamma,\gamma} \end{bmatrix} \mathcal{R}_3(X')[t] + \begin{bmatrix} I_{d_x} & & & \\ & 1 & & \\ & 1 & 1 & \\ & & & O_{\gamma,\gamma} \end{bmatrix} X'[t+1], \tag{150}$$

and attain

$$\mathcal{R}_3\mathcal{N}_1\mathcal{R}_2\mathcal{R}_1\mathcal{P}(X)[t] = \begin{bmatrix} X[t]_{1:d_x} + t\mathbf{1}_{d_x} \\ \mathcal{N}_{l,MLP}\left(X[t]_{1:d_x} + t\mathbf{1}_{d_x}\right)_{d_x+1} \\ \sum_{j=1}^{t} \mathcal{N}_{l,MLP}\left(X[j]_{1:d_x} + j\mathbf{1}_{d_x}\right)_{d_x+1} + \sum_{j=1}^{t} X[j]_{d_x+1} \\ \mathbf{0}_\gamma \end{bmatrix}. \quad (151)$$

With the token-wise projection map $\mathcal{Q} : \mathbb{R}^{d_x+2+\gamma} \to \mathbb{R}^{d_x+1}$ defined by $\mathcal{Q}(X') = \begin{bmatrix} X'_{1:d_x} \\ X'_{d_x+2} \end{bmatrix}$, an RNN $\mathcal{Q}\mathcal{R}_3\mathcal{N}_l\mathcal{R}_2\mathcal{R}_1\mathcal{P} : \mathbb{R}^{(d_x+1)\times N} \to \mathbb{R}^{(d_x+1)\times N}$ of width $d_x+2+\gamma$ maps $X \in \mathbb{R}^{(d_x+1)\times N}$ to

$$\mathcal{Q}\mathcal{R}_3\mathcal{N}_l\mathcal{R}_2\mathcal{R}_1\mathcal{P}(X)[t] = \begin{bmatrix} X[t]_{1:d_x} + t\mathbf{1}_{d_x} \\ \sum_{j=1}^{t} \mathcal{N}_{l,MLP}\left(X[j]_{1:d_x} + j\mathbf{1}_{d_x}\right)_{d_x+1} + \sum_{j=1}^{t} X[j]_{d_x+1} \end{bmatrix}. \quad (152)$$

Since $\mathcal{N}_{l,MLP}\left(X[j]_{1:d_x} + j\mathbf{1}_{d_x}\right)_{d_x+1} \to b_l[j]X[j]_{1:d_x}$, we have

$$\sup_{X\in(K\times K')^N} \|\mathcal{T}\mathcal{R}_l(X) - \mathcal{Q}\mathcal{R}_3\mathcal{N}_l\mathcal{R}_2\mathcal{R}_1\mathcal{P}(X)\| \to 0, \quad (153)$$

as $\delta_l \to 0$. Approximating all $\mathcal{T}\mathcal{R}_l$ in Appendix D finishes the proof.

Appendix F. Proof of Lemma 19

The main idea of the proof is to separate the linear sum $\sum_{j=1}^{N} A_j[t]x[j]$ into the past-dependent part $\sum_{j=1}^{t-1} A_j[t]x[j]$ and the remainder part $\sum_{j=t}^{N} A_j[t]x[j]$. Then, we construct modified TBRNN with $2N$ cells; the former N cells have only a forward recurrent cell to compute the past-dependent part, and the latter N cells have only a backward recurrent cell to compute the remainder.

Let the first N modified TRNN cells $\mathcal{R}_l : \mathbb{R}^{(d_x+1)\times N} \to \mathbb{R}^{(d_x+1)\times N}$ for $1 \le l \le N$ be defined as in the proof of Lemma 11:

$$\mathcal{R}_l\left(X\right)[t+1] = A_l\mathcal{R}_l\left(X\right)[t] + B_l[t]X[t+1], \quad (154)$$

where $A_l = \begin{bmatrix} O_{d_x,d_x} & O_{d_x,1} \\ O_{1,d_x} & 1 \end{bmatrix}$, $B_l[t] = \begin{bmatrix} I_{d_x} & O_{d_x,1} \\ b_l[t] & 1 \end{bmatrix}$ for $b_l[t] \in \mathbb{R}^{1\times d_x}$. Then, with token-wise lifting map $\mathcal{P} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x+1}$ defined by $\mathcal{P}(x) = \begin{bmatrix} x \\ 0 \end{bmatrix}$, we construct modified TRNN $\mathcal{N} : \mathcal{R}_N \circ \cdots \circ \mathcal{R}_1 \circ \mathcal{P} : \mathbb{R}^{d_x\times N} \to \mathbb{R}^{(d_x+1)\times N}$. We know that if $C_i[m] \in \mathbb{R}^{1\times d_x}$ are given for $1 \le m \le N$ and $1 \le i \le m$, there exist $b_l[t]$ for $1 \le l \le N$, such that

$$\mathcal{N}_N(x)[m] = \begin{bmatrix} x[m] \\ \sum_{i=1}^{m} C_i[m]x[i] \end{bmatrix}. \quad (155)$$

Therefore, we will determine $C_i[m]$ after constructing the latter N cells. Let $f_m = \sum_{i=1}^{m} C_i[m]x[i]$ for brief notation.

After $\mathcal{N}_N$, construct N modified TRNN cells $\bar{\mathcal{R}}_l : \mathbb{R}^{(d_x+1)\times N} \to \mathbb{R}^{(d_x+1)\times N}$ for $1 \le l \le N$ in reverse order:

$$\bar{\mathcal{R}}_l\left(\bar{X}\right)[t-1] = \bar{A}_l\bar{\mathcal{R}}_l\left(\bar{X}\right)[t] + \bar{B}_l[t]\bar{X}[t-1], \quad (156)$$

where $\bar{A}_l = \begin{bmatrix} O_{d_x,d_x} & O_{d_x,1} \\ O_{1,d_x} & 1 \end{bmatrix}$, $\bar{B}_l[t] = \begin{bmatrix} I_{d_x} & O_{d_x,1} \\ \bar{b}_l[t] & 1 \end{bmatrix}$ for $\bar{b}_l[t] \in \mathbb{R}^{1\times d_x}$. Define $\bar{\mathcal{N}}_N = \bar{\mathcal{R}}_N \circ \cdots \circ \bar{\mathcal{R}}_1$, and we obtain the following result after a similar calculation with input sequence $\bar{X}[t] = \mathcal{N}_N(x)[t] = \begin{bmatrix} x[t] \\ f_t \end{bmatrix}$:

$$\bar{\mathcal{N}}_N\left(\bar{X}\right)[N+1-t] \tag{157}$$

$$= \begin{bmatrix} x[N+1-t] \\ \sum_{j=1}^{t}\left[\sum_{i=1}^{N}\binom{N+t-i-j}{N-i}\bar{b}_i[N+1-j]x[N+1-j] + \binom{N+t-1-j}{N-1}f_{N+1-j}\right] \end{bmatrix}. \tag{158}$$

We want to find f_m and $\bar{b}_i[m]$ so that

$$\bar{\mathcal{N}}_N\left(\bar{X}\right)[N+1-t]_{d_x+1} = \sum_{i=1}^{N} A_i[N+1-t]x[i], \tag{159}$$

for each $t = 1, 2, \ldots, N$.

Note that $\sum_{j=1}^{t}\sum_{i=1}^{N}\binom{N+t-i-j}{N-i}\bar{b}_i[N+1-j]x[N+1-j]$ does not contain $x[1], x[2], \ldots, x[N-t]$ terms, so $\sum_{j=1}^{t}\binom{N+t-1-j}{N-1}f_{N+1-j}$ should contain $\sum_{i=1}^{N-t} A_i[N+1-t]x[i]$.

$$\sum_{j=1}^{t}\binom{N+t-1-j}{N-1}f_{N+1-j} = \sum_{j=1}^{t}\binom{N+t-1-j}{N-1}\sum_{i=1}^{N+1-j} C_i[N+1-j]x[i] \tag{160}$$

$$= \sum_{j=1}^{t}\sum_{i=1}^{N+1-j}\binom{N+t-1-j}{N-1}C_i[N+1-j]x[i] \tag{161}$$

$$= \sum_{i=N+2-t}^{N}\sum_{j=1}^{N+1-i}\binom{N+t-1-j}{N-1}C_i[N+1-j]x[i] \tag{162}$$

$$+ \sum_{i=1}^{N+1-t}\sum_{j=1}^{t}\binom{N+t-1-j}{N-1}C_i[N+1-j]x[i]. \tag{163}$$

Since matrix $\Lambda_i = \left\{\binom{N+t-1-j}{N-1}\right\}_{1\le t\le N+1-i, 1\le j\le N+1-i}$ is a lower triangular $(N+1-i)\times(N+1-i)$ matrix with unit diagonal components, there exist $C_i[i], C_i[i+1], \ldots, C_i[N]$ such that

$$\sum_{j=1}^{t}\binom{N+t-1-j}{N-1}C_i[N+1-j] = A_i[N+1-t], \tag{164}$$

for each $t = 1, 2, \ldots, N+1-i$.

We now have

$$\sum_{j=1}^{t} \binom{N+t-1-j}{N-1} f_{N+1-j} \tag{165}$$

$$= \sum_{i=N+2-t}^{N} \sum_{j=1}^{N+1-i} \binom{N+t-1-j}{N-1} C_i[N+1-j]x[i] + \sum_{i=1}^{N+1-t} A_i[N+1-t]x[i] \tag{166}$$

$$= \sum_{i=1}^{t-1} \sum_{j=1}^{i} \binom{N+t-1-j}{N-1} C_{N+1-i}[N+1-j]x[N+1-i] + \sum_{i=1}^{N+1-t} A_i[N+1-t]x[i] \tag{167}$$

$$= \sum_{j=1}^{t-1} \sum_{i=1}^{j} \binom{N+t-1-i}{N-1} C_{N+1-j}[N+1-i]x[N+1-j] + \sum_{i=1}^{N+1-t} A_i[N+1-t]x[i]. \tag{168}$$

We switch i and j for the last equation. By Corollary 26, there exist $\bar{b}_i[N+1-j]$ satisfying

$$\sum_{i=1}^{N} \binom{N+t-i-j}{N-i} \bar{b}_i[N+1-j] \tag{169}$$

$$= A_{N+1-j}[N+1-t] - \sum_{i=1}^{j} \binom{N+t-1-i}{N-1} C_{N+1-j}[N+1-i], \tag{170}$$

for $j = 1, 2, \ldots, t-1$, and

$$\sum_{i=1}^{N} \binom{N+t-i-j}{N-i} \bar{b}_i[N+1-j] \tag{171}$$

$$= A_{N+1-j}[N+1-t], \tag{172}$$

for $j = t$.
With the above $C_i[m]$ and $\bar{b}_i[m]$, equation (159) holds for each $t = 1, 2, \ldots, N$. It remains to construct modified TRNN cells to implement f_m, which comes directly from the proof of Lemma 11.

References

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728. PMLR, 2016.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34: 3965–3977, 2021.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772. PMLR, 2014.

Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

Joshua Hanson and Maxim Raginsky. Universal simulation of stable dynamical systems by recurrent neural nets. In *Learning for Dynamics and Control*, pages 384–392. PMLR, 2020.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

Jesse Johnson. Deep, skinny neural networks are not universal approximators. In *International Conference on Learning Representations*, 2019.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on learning theory*, pages 2306–2327. PMLR, 2020.

PM Lavanya and E Sasikala. Deep learning techniques on text classification using natural language processing (nlp) in social healthcare network: A comprehensive survey. In *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, pages 603–609. IEEE, 2021.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

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.

Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *Advances in Neural Information Processing Systems*, 31, 2018.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. *arXiv preprint arXiv:2006.08859*, 2020.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *International Conference on Learning Representations*, 2021.

David Rolnick and Max Tegmark. The power of deeper networks for expressing natural functions. *In the International Conference on Learning Representations*, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Anton Maximilian Schäfer and Hans-Georg Zimmermann. Recurrent neural networks are universal approximators. *International journal of neural systems*, 17(04):253–263, 2007.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503, 2017.

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

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2):787–794, 2020.