A DEEP RECURRENT NEURAL NETWORK VIA UNFOLDING REWEIGHTED ℓ_1-ℓ_1 MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep unfolding methods design deep neural networks as learned variations of optimization methods. These networks have been shown to achieve faster convergence and higher accuracy than the original optimization methods. In this line of research, this paper develops a novel deep recurrent neural network (coined reweighted-RNN) by unfolding a reweighted ℓ_1-ℓ_1 minimization algorithm and applies it to the task of sequential signal reconstruction. To the best of our knowledge, this is the first deep unfolding method that explores reweighted minimization. Due to the underlying reweighted minimization model, our RNN has a different soft-thresholding function (alias, different activation function) for each hidden unit in each layer. Furthermore, it has higher network expressivity than existing deep unfolding RNN models due to the over-parameterizing weights. Moreover, we establish theoretical generalization error bounds for the proposed reweighted-RNN model by means of Rademacher complexity. The bounds reveal that the parameterization of the proposed reweighted-RNN ensures good generalization. We apply the proposed reweighted-RNN to the problem of video-frame reconstruction from low-dimensional measurements, that is, sequential frame reconstruction. The experimental results on the moving MNIST dataset demonstrate that the proposed deep reweighted-RNN significantly outperforms existing RNN models.

1 INTRODUCTION

The problem of reconstructing sequential signals from low-dimensional measurements across time is of great importance for a number of applications such as time-series data analysis, future-frame prediction, and compressive video sensing. Specifically, we consider the problem of reconstructing a sequence of signals $\mathbf{s}_t \in \mathbb{R}^{n_0}$, $t = 1, 2, \ldots, T$, from low-dimensional measurements $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t$, where $\mathbf{A} \in \mathbb{R}^{n \times n_0}$ $(n \ll n_0)$ is a sensing matrix. We assume that $\mathbf{s}_t$ has a sparse representation $\mathbf{h}_t \in \mathbb{R}^h$ in a dictionary $\mathbf{D} \in \mathbb{R}^{n_0 \times h}$, that is, $\mathbf{s}_t = \mathbf{D}\mathbf{h}_t$. At each time step t, the signal $\mathbf{s}_t$ can be independently reconstructed using the measurements $\mathbf{x}_t$ by solving (Donoho, 2006):

$$\min_{\mathbf{h}_t} \Big\{ \frac{1}{2} \|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2 + \lambda \|\mathbf{h}_t\|_1 \Big\}, \tag{1}$$

where $\|\cdot\|_p$ is the ℓ_p-norm and λ is a regularization parameter. The iterative shrinkage-thresholding algorithm (ISTA) (Daubechies et al., 2004) solves (1) by iterating over $\mathbf{h}_t^{(l)} = \phi_{\frac{\lambda}{c}}(\mathbf{h}_t^{(l-1)} - \frac{1}{c}\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{D}\mathbf{h}_t^{(l-1)} - \mathbf{x}_t))$, where l is the iteration counter, $\phi_\gamma(u) = \mathrm{sign}(u)[0, |u| - \gamma]_+$ is the soft-thresholding operator, $\gamma = \frac{\lambda}{c}$, and c is an upper bound on the Lipschitz constant of the gradient of $\frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2$.

Deep neural networks (DNN) have achieved state-of-the-art performance in solving (1) for individual signals, both in terms of accuracy and inference speed (Mousavi et al., 2015). However, these models are often criticized for their lack of interpretability and theoretical guarantees (Lucas et al., 2018). Motivated by this, several studies focus on designing DNNs that incorporate domain knowledge, namely, signal priors. These include deep unfolding methods which design neural networks to learn approximations of iterative optimization algorithms. Examples of this approach are LISTA (Gregor & LeCun, 2010) and its variants, including ADMM-Net (Sun et al., 2016), AMP (Borgerding et al., 2017), and an unfolded version of the iterative hard thresholding algorithm (Xin et al., 2016).

LISTA (Gregor & LeCun, 2010) unrolls the iterations of ISTA into a feed-forward neural network with weights, where each layer implements an iteration: $\mathbf{h}_t^{(l)} = \phi_{\gamma^{(l)}}(\mathbf{W}^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{U}^{(l)}\mathbf{x}_t)$, with $\mathbf{W}^{(l)} = \mathbf{I} - \frac{1}{c}\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}$, $\mathbf{U}^{(l)} = \frac{1}{c}\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$, and $\gamma^{(l)}$ being learned from data. It has been shown (Gregor & LeCun, 2010; Sprechmann et al., 2015) that a d-layer LISTA network with trainable parameters $\boldsymbol{\Theta} = \{\mathbf{W}^{(l)}, \mathbf{U}^{(l)}, \gamma^{(l)}\}_{l=1}^{d}$ achieves the same performance as the original ISTA but with much fewer iterations (i.e., number of layers). Recent studies (Chen et al., 2018; Liu et al., 2019) have found that exploiting dependencies between $\mathbf{W}^{(l)}$ and $\mathbf{U}^{(l)}$ leads to reducing the number of trainable parameters while retaining the performance of LISTA. These works provided theoretical insights to the convergence conditions of LISTA. However, the problem of *designing deep unfolding methods for dealing with sequential signals* is significantly less explored.

Under the assumption that sequential signal instances are correlated, we consider the following sequential signal reconstruction problem:

$$\min_{\mathbf{h}_t} \Big\{ \frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2 + \lambda_1\|\mathbf{h}_t\|_1 + \lambda_2 R(\mathbf{h}_t, \mathbf{h}_{t-1}) \Big\}, \tag{2}$$

where $\lambda_1, \lambda_2 > 0$ are regularization parameters and $R(\mathbf{h}_t, \mathbf{h}_{t-1})$ is an added regularization term that expresses the similarity of the representations $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$ of two consecutive signals. Wisdom et al. (2017) proposed an RNN design (coined Sista-RNN) by unfolding the sequential version of ISTA. That study assumed that two consecutive signals are close in the ℓ_2-norm sense, formally, $R(\mathbf{h}_t, \mathbf{h}_{t-1}) = \frac{1}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1}\|_2^2$, where $\mathbf{F} \in \mathbb{R}^{n_0 \times n_0}$ is a correlation matrix between $\mathbf{s}_t$ and $\mathbf{s}_{t-1}$. More recently, the study by Le et al. (2019) designed the ℓ_1-ℓ_1-RNN, which stems from unfolding an algorithm that solves the ℓ_1-ℓ_1 minimization problem (Mota et al., 2017; 2016). This is a version of Problem (2) with $R(\mathbf{h}_t, \mathbf{h}_{t-1}) = \|\mathbf{h}_t - \mathbf{G}\mathbf{h}_{t-1}\|_1$, where $\mathbf{G} \in \mathbb{R}^{h \times h}$ is an affine transformation that promotes the correlation between $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$. Both studies (Wisdom et al., 2017; Le et al., 2019) have shown that carefully-designed deep RNN models outperform the generic RNN model and ISTA (Daubechies et al., 2004) in the task of sequential frame reconstruction.

It has been shown that reweighted algorithms—such as the reweighted ℓ_1 minimization method by Candès et al. (2008) and the reweighted ℓ_1-ℓ_1 minimization by Luong et al. (2018)—outperform their non-reweighted counterparts. Driven by this observation, this paper proposes a novel deep RNN architecture by unfolding a reweighted-ℓ_1-ℓ_1 minimization algorithm. Due to the reweighting, our network has higher expressivity than existing RNN models leading to better data representations, especially when depth increases. This is in line with recent studies (He et al., 2016; Cortes et al.; Huang et al., 2017), which have shown that better performance can be achieved by highly over-parameterized networks, i.e., networks with far more parameters than the number of training samples. While the most recent studies (related over-parameterized DNNs) consider fully-connected networks applied on classification problems (Neyshabur et al., 2019), our approach focuses on deep-unfolding architectures and opts to understand how the networks learn a low-complexity representation for sequential signal reconstruction, which is a regression problem across time. Furthermore, while there have been efforts to build deep RNNs (Pascanu et al., 2014; Li et al., 2018; Luo et al., 2017; Wisdom et al., 2017), examining the generalization property of such deep RNN models on unseen sequential data still remains elusive. In this work, we derive the generalization error bound of the proposed design and further compare it with existing RNN bounds (Zhang et al., 2018; Kusupati et al., 2018).

Contributions. The contributions of this work are as follows:

- We propose a principled deep RNN model for sequential signal reconstruction by unfolding a reweighted ℓ_1-ℓ_1 minimization method. Our reweighted-RNN model employs different soft-thresholding functions that are adaptively learned per hidden unit. Furthermore, the proposed model is over-parameterized, has high expressivity and can be efficiently stacked.
- We derive the generalization error bound of the proposed model (and deep RNNs) by measuring Rademacher complexity and show that the over-parameterization of our RNN ensures good generalization. To best of our knowledge, this is the first generalization error bound for deep RNNs; moreover, our bound is tighter than existing bounds derived for shallow RNNs (Zhang et al., 2018; Kusupati et al., 2018).
- We provide experiments in the task of reconstructing video sequences from low-dimensional measurements. We show significant gains when using our model compared to several state-of-the-art RNNs (including unfolding architectures), especially when the depth of RNNs increases.

2 A DEEP RNN VIA UNFOLDING REWEIGHTED-ℓ_1-ℓ_1 MINIMIZATION

In this section, we describe a reweighted ℓ_1-ℓ_1 minimization problem for sequential signal reconstruction and propose an iterative algorithm based on the proximal method. We then design a deep RNN architecture by unfolding this algorithm.

The proposed reweighted ℓ_1-ℓ_1 minimization. We introduce the following problem:

$$\min_{\mathbf{h}_t} \Big\{ \frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{Z}\mathbf{h}_t\|_2^2 + \lambda_1 \|\mathbf{g} \circ \mathbf{Z}\mathbf{h}_t\|_1 + \lambda_2 \|\mathbf{g} \circ (\mathbf{Z}\mathbf{h}_t - \mathbf{G}\mathbf{h}_{t-1})\|_1 \Big\}, \tag{3}$$

where "$\circ$" denotes element-wise multiplication, $\mathbf{g} \in \mathbb{R}^h$ is a vector of positive weights, $\mathbf{Z} \in \mathbb{R}^{h \times h}$ is a reweighting matrix, and $\mathbf{G} \in \mathbb{R}^{h \times h}$ is an affine transformation that promotes the correlation between $\mathbf{h}_{t-1}$ and $\mathbf{h}_t$. Intuitively, $\mathbf{Z}$ is adopted to transform $\mathbf{h}_t$ to $\mathbf{Z}\mathbf{h}_t \in \mathbb{R}^h$, producing a reweighted version of it. Thereafter, $\mathbf{g}$ aims to reweight each transformed component of $\mathbf{Z}\mathbf{h}_t$ and $\mathbf{Z}\mathbf{h}_t - \mathbf{G}\mathbf{h}_{t-1}$ in the ℓ_1-norm regularization terms. Because of applying reweighting (Candès et al., 2008), the solution of Problem (3) is a more accurate sparse representation compared to the solution of the ℓ_1-ℓ_1 minimization problem in Le et al. (2019) (where $\mathbf{Z} = \mathbf{I}$ and $\mathbf{g} = \mathbf{I}$). Furthermore, the use of the reweighting matrix $\mathbf{Z}$ to transform $\mathbf{h}_t$ to $\mathbf{Z}\mathbf{h}_t$ differentiates Problem (3) from the reweighted ℓ_1-ℓ_1 minimization problem in Luong et al. (2018) where $\mathbf{Z} = \mathbf{I}$.

The objective function in (3) consists of the differentiable fidelity term $f(\mathbf{Z}\mathbf{h}_t) = \frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{Z}\mathbf{h}_t\|_2^2$ and the non-smooth term $g(\mathbf{Z}\mathbf{h}_t) = \lambda_1 \|\mathbf{g} \circ \mathbf{Z}\mathbf{h}_t\|_1 + \lambda_2 \|\mathbf{g} \circ (\mathbf{Z}\mathbf{h}_t - \mathbf{G}\mathbf{h}_{t-1})\|_1$. We use a proximal gradient method (Beck & Teboulle, 2009) to solve (3): At iteration l, we first update $\mathbf{h}_t^{(l-1)}$—after being multiplied by $\mathbf{Z}_l$—with a gradient descent step on the fidelity term as $\mathbf{u} = \mathbf{Z}_l \mathbf{h}_t^{(l-1)} - \frac{1}{c}\mathbf{Z}_l \nabla f(\mathbf{h}_t^{(l-1)})$, where $\nabla f(\mathbf{h}_t^{(l-1)}) = \mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{D}\mathbf{h}_t^{(l-1)} - \mathbf{x}_t)$. Then, $\mathbf{h}_t^{(l)}$ is updated as

$$\mathbf{h}_t^{(l)} = \Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \mathbf{G}\mathbf{h}_{t-1}} \Big(\mathbf{Z}_l \mathbf{h}_t^{(l)} - \frac{1}{c}\mathbf{Z}_l \nabla f(\mathbf{h}_t^{(l)}) \Big), \tag{4}$$

where the proximal operator $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \mathbf{G}\mathbf{h}_{t-1}}(\mathbf{u})$ is defined as

$$\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \boldsymbol{\hbar}}(\mathbf{u}) = \arg\min_{\mathbf{v} \in \mathbb{R}^h} \Big\{ \frac{1}{c} g(\mathbf{v}) + \frac{1}{2}||\mathbf{v} - \mathbf{u}||_2^2 \Big\}, \tag{5}$$

with $\boldsymbol{\hbar} = \mathbf{G}\mathbf{h}_{t-1}$. Since the minimization problem is separable, we can minimize (5) independently for each of the elements g_l, $\hbar$, u of the corresponding $\mathbf{g}_l$, $\boldsymbol{\hbar}$, $\mathbf{u}$ vectors. After solving (5), we obtain $\Phi_{\frac{\lambda_1}{c}\mathrm{g}_l, \frac{\lambda_2}{c}\mathrm{g}_l, \hbar}(u)$ [for solving (5), we refer to Proposition B.1 in Appendix B]. For $\hbar \geq 0$:

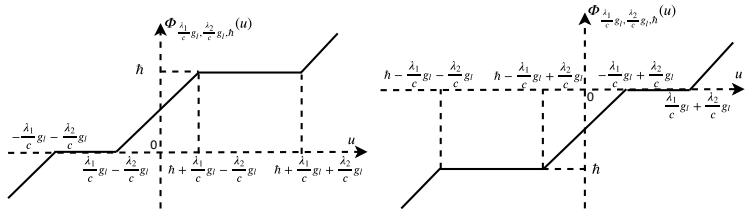
$$\Phi_{\frac{\lambda_1}{c}\mathrm{g}_l, \frac{\lambda_2}{c}\mathrm{g}_l, \hbar \geq 0}(u) = \begin{cases} u - \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l, & \hbar + \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l < u < \infty \\ \hbar, & \hbar + \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l \leq u \leq \hbar + \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l \\ u - \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l, & \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l < u < \hbar + \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l \\ 0, & -\frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l \leq u \leq \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l \\ u + \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l, & -\infty < u < -\frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l, \end{cases} \tag{6}$$

and for $\hbar < 0$:

$$\Phi_{\frac{\lambda_1}{c}\mathrm{g}_l, \frac{\lambda_2}{c}\mathrm{g}_l, \hbar < 0}(u) = \begin{cases} u - \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l, & \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l < u < \infty \\ 0, & -\frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l \leq u \leq \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l \\ u + \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l, & \hbar - \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l < u < -\frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l \\ \hbar, & \hbar - \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l \leq u \leq \hbar - \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l \\ u - \frac{\lambda_1}{c}\mathrm{g}_l + \frac{\lambda_2}{c}\mathrm{g}_l, & -\infty < u < \hbar - \frac{\lambda_1}{c}\mathrm{g}_l - \frac{\lambda_2}{c}\mathrm{g}_l \end{cases} \tag{7}$$

Fig. 1 depicts the proximal operators for $\hbar \geq 0$ and $\hbar < 0$. Observe that different values of g_l lead to different shapes of the proximal functions $\Phi_{\frac{\lambda_1}{c}\mathrm{g}_l, \frac{\lambda_2}{c}\mathrm{g}_l, \hbar}(u)$ for each element u of $\mathbf{u}$.

Our iterative algorithm is given in Algorithm 1. We reconstruct a sequence $\mathbf{h}_1, \ldots, \mathbf{h}_T$ from a sequence of measurements $\mathbf{x}_1, \ldots, \mathbf{x}_T$. For each time step t, Step 6 applies a gradient descent update for $f(\mathbf{Z}\mathbf{h}_{t-1})$ and Step 7 applies the proximal operator $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \mathbf{G}\mathbf{h}_{t-1}^{(d)}}$ element-wise to the result. Let us compare the proposed method against the algorithm in Le et al. (2019)—which resulted in the



(a) $\hbar \geq 0$. (b) $\hbar < 0$.

Figure 1: The generic form of the proximal operators for Algorithm 1 - but also the activation function in the proposed reweighted-RNN. Note that per unit per layer g_l leads to a different activation function.

Algorithm 1: The proposed algorithm for sequential signal reconstruction.

1 **Input:** Measurements $\mathbf{x}_1, \ldots, \mathbf{x}_T$, measurement matrix $\mathbf{A}$, dictionary $\mathbf{D}$, affine transform $\mathbf{G}$, initial $\mathbf{h}_0^{(d)} \equiv \mathbf{h}_0$, reweighting matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_d$ and vectors $\mathbf{g}_1, \ldots, \mathbf{g}_d$, c, λ_1, λ_2.
2 **Output:** Sequence of sparse codes $\mathbf{h}_1, \ldots, \mathbf{h}_T$.
3 **for** *t = 1,...,T* **do**
4 $\quad \mathbf{h}_t^{(0)} = \mathbf{G}\mathbf{h}_{t-1}^{(d)}$
5 $\quad$ **for** *l = 1 to d* **do**
6 $\quad\quad \mathbf{u} = [\mathbf{Z}_l - \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}]\mathbf{h}_t^{(l-1)} + \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{x}_t$
7 $\quad\quad \mathbf{h}_t^{(l)} = \Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \mathbf{G}\mathbf{h}_{t-1}^{(d)}}(\mathbf{u})$
8 $\quad$ **end**
9 **end**
10 **return** $\mathbf{h}_1^{(d)}, \ldots, \mathbf{h}_T^{(d)}$

ℓ_1-ℓ_1-RNN—that solves the ℓ_1-ℓ_1 minimization in Mota et al. (2016) (where $\mathbf{Z}_l = \mathbf{I}$ and $\mathbf{g}_l = \mathbf{I}$). In that algorithm, the update terms in Step 6, namely $\mathbf{I} - \frac{1}{c}\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}$ and $\frac{1}{c}\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$, and the proximal operator in Step 7 are the same for all iterations of l. In contrast, Algorithm 1 uses a different $\mathbf{Z}_l$ matrix per iteration to reparameterize the update terms (Step 6) and, through updating $\mathbf{g}_l$, it applies a different proximal operator to each element $\mathbf{u}$ (in Step 7) per iteration l.

The proposed reweighted-RNN. We now describe the proposed architecture for sequential signal recovery, designed by unrolling the steps of Algorithm 1 across the iterations $l = 1, \ldots, d$ (yielding the hidden layers) and time steps $t = 1, \ldots, T$. Specifically, the l-th hidden layer is given by

$$\mathbf{h}_t^{(l)} = \begin{cases} \Phi_{\frac{\lambda_1}{c}\mathbf{g}_1, \frac{\lambda_2}{c}\mathbf{g}_1, \mathbf{G}\mathbf{h}_{t-1}^{(d)}}\left(\mathbf{W}_1\mathbf{h}_{t-1}^{(d)} + \mathbf{U}_1\mathbf{x}_t\right), & \text{if } l = 1, \\ \Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \mathbf{G}\mathbf{h}_{t-1}^{(d)}}\left(\mathbf{W}_l\mathbf{h}_t^{(l-1)} + \mathbf{U}_l\mathbf{x}_t\right), & \text{if } l > 1, \end{cases} \tag{8}$$

and the reconstructed signal at time step t is given by $\hat{\mathbf{s}}_t = \mathbf{D}\mathbf{h}_t^{(d)}$; where $\mathbf{U}_l$, $\mathbf{W}_l$, $\mathbf{V}$ are defined as

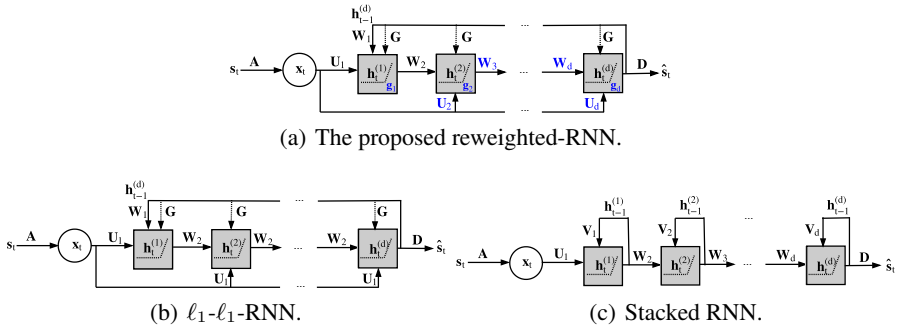
$$\mathbf{U}_l = \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}, \forall l, \tag{9}$$

$$\mathbf{W}_1 = \mathbf{Z}_1\mathbf{G} - \frac{1}{c}\mathbf{Z}_1\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}\mathbf{G}, \tag{10}$$

$$\mathbf{W}_l = \mathbf{Z}_l - \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}, \; l > 1. \tag{11}$$

The activation function is the proximal operator $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \boldsymbol{\hbar}}(\mathbf{u})$ with learnable parameters λ_1, λ_2, c, $\mathbf{g}_l$ (see Fig. 1 for the shapes of the activation functions).

Fig. 2(a) depicts the architecture of the proposed reweighted-RNN. Input vectors $\mathbf{s}_t$, $t = 1, \ldots, T$ are compressed by a linear measurement layer $\mathbf{A}$, resulting in compressive measurements $\mathbf{x}_t$. The reconstructed vectors $\hat{\mathbf{s}}_t$, $t = 1, \ldots, T$, are obtained by multiplying linearly the hidden representation $\mathbf{h}_t^{(d)}$ with the dictionary $\mathbf{D}$. We train our network in an end-to-end fashion. During training, we minimize



(a) The proposed reweighted-RNN.

(b) ℓ_1-ℓ_1-RNN.

(c) Stacked RNN.

Figure 2: The proposed (a) reweighted-RNN vs. (b) ℓ_1-ℓ_1-RNN and (c) Stacked RNN.

the loss function $\mathcal{L}(\boldsymbol{\Theta}) = \mathop{\mathbb{E}}_{\mathbf{s}_1,\cdots,\mathbf{s}_T}\left[\sum_{t=1}^{T} \|\mathbf{s}_t - \hat{\mathbf{s}}_t\|_2^2\right]$ using stochastic gradient descent on mini-batches, where the trainable parameters are $\boldsymbol{\Theta} = \{\mathbf{A}, \mathbf{D}, \mathbf{G}, \mathbf{h}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_d, \mathbf{g}_1, \ldots, \mathbf{g}_d, c, \lambda_1, \lambda_2\}$.

We now compare the proposed reweighted-RNN [Fig. 2(a)] against the recent ℓ_1-ℓ_1-RNN (Le et al., 2019) [Fig. 2(b)]. The l-th hidden layer in ℓ_1-ℓ_1-RNN is given by

$$\mathbf{h}_t^{(l)} = \begin{cases} \varPhi_{\frac{\lambda_1}{c}, \frac{\lambda_2}{c}, \mathbf{G}\mathbf{h}_{t-1}^{(d)}}\left(\mathbf{W}_1\mathbf{h}_{t-1}^{(d)} + \mathbf{U}_1\mathbf{x}_t\right), \text{ if } l = 1, \\ \varPhi_{\frac{\lambda_1}{c}, \frac{\lambda_2}{c}, \mathbf{G}\mathbf{h}_{t-1}^{(d)}}\left(\mathbf{W}_2\mathbf{h}_t^{(l-1)} + \mathbf{U}_1\mathbf{x}_t\right), \text{ if } l > 1. \end{cases} \tag{12}$$

The proposed model has the following advantages over ℓ_1-ℓ_1-RNN. Firstly, ℓ_1-ℓ_1-RNN uses the proximal operator $\varPhi_{\frac{\lambda_1}{c}, \frac{\lambda_2}{c}, \boldsymbol{\hbar}}(\mathbf{u})$ as activation function, whose learnable parameters λ_1, λ_2 are fixed across the network. Conversely, the corresponding parameters $\frac{\lambda_1}{c}\mathbf{g}_l$ and $\frac{\lambda_2}{c}\mathbf{g}_l$ [see (6), (7), and Fig. 1] in our proximal operator, $\varPhi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \boldsymbol{\hbar}}(\mathbf{u})$, are learned for each hidden layer due to the reweighting vector $\mathbf{g}_l$; hence, the proposed model has a different activation function for each unit per layer. The second difference comes from the set of parameters $\{\mathbf{W}_l, \mathbf{U}_l\}$ in (12) and (8). The ℓ_1-ℓ_1-RNN model uses the same $\{\mathbf{W}_2, \mathbf{U}_1\}$ for the second and higher layers. In contrast, our reweighted-RNN has different sets of $\{\mathbf{W}_l, \mathbf{U}_l\}$ per hidden layer due to the reweighting matrix $\mathbf{Z}_l$. These two aspects [which are schematically highlighted in blue fonts in Fig. 2(a)] can lead to an increase in the learning capability of the proposed reweighted-RNN, especially when the depth of the model increases.

In comparison to a generic stacked RNN (Pascanu et al., 2014) [Fig. 2(c)], reweighted-RNN promotes the inherent data structure, that is, each vector $\mathbf{s}_t$ has a sparse representation $\mathbf{h}_t$ and consecutive $\mathbf{h}_t$'s are correlated. This design characteristic of the reweighted-RNN leads to residual connections which reduce the risk of vanishing gradients during training [the same idea has been shown in several works (He et al., 2016; Huang et al., 2017) in deep neural network literature]. Furthermore, in (9) and (11), we see a weight coupling of $\mathbf{W}_l$ and $\mathbf{U}_l$ (due to the shared components of $\mathbf{A}$, $\mathbf{D}$ and $\mathbf{Z}$). This coupling satisfies the necessary condition of the convergence in Chen et al. (2018) (Theorem 1). Using Theorem 2 in Chen et al. (2018), it can be shown that reweighted-RNN, in theory, needs a smaller number of iterations (i.e., d in Algorithm 1) to reach convergence, compared to ISTA (Daubechies et al., 2004) and FISTA (Beck & Teboulle, 2009).

3 GENERALIZATION ERROR BOUND

While increasing the network expressivity, the over-parameterization of reweighted-RNN raises the question of whether our network ensures good generalization. In this section, we derive and analyze the generalization properties of the proposed reweighted-RNN model in comparison to state-of-the-art RNN architectures. We provide bounds on the Rademacher complexity (Shalev-Shwartz & Ben-David, 2014) for functional classes of the considered deep RNNs, which are used to derive generalization error bounds for evaluating their generalization properties (we refer to Appendix C.1 for definitions of the Rademacher complexity and the generalization error bound).

Preliminaries: We consider a deep RNN as a d-layer network $f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d)} \in \mathcal{F}_{d,T} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}^h$ with weight parameters $\boldsymbol{\mathcal{W}} = (\mathbf{W}_1, ..., \mathbf{W}_d)$ and $\boldsymbol{\mathcal{U}} = (\mathbf{U}_1, ..., \mathbf{U}_d)$, where $\mathbf{W}_l \in \mathbb{R}^{h\times h}$, $\mathbf{U}_l \in$

$\mathbb{R}^{h\times n}$. As in Bartlett et al. (2017); Golowich et al. (2018); Neyshabur et al. (2019; 2015), we derive generalization error bounds by controlling norms of the weight matrices. Let $\|\mathbf{W}_l\|_{p,q} = (\sum_j (\|\mathbf{w}_{l,j}\|_p)^q)^{1/q}$ define the $\ell_{p,q}$-norm, $p, q \geq 1$, of the weight-matrix $\mathbf{W}_l$, where $\mathbf{w}_{l,j} \in \mathbb{R}^h$ is the jth row of $\mathbf{W}_l$. Since we focus on deep networks with soft-thresholding-based activation units—designed by unfolding algorithms for ℓ_1-norm minimization—we derive the network complexities under bounding per-unit ℓ_1 regularization, i.e., $\|\mathbf{w}_{l,j}\|_1$. We also denote $\|\mathbf{W}_l\|_{1,\infty} = \max_j \|\mathbf{w}_{l,j}\|_1$ as the maximum of the ℓ_1-norms of the matrix's rows. We assume that the ℓ_1-norm of the weights of each neuron is bounded as $\|\mathbf{W}_l\|_{1,\infty} = \max_j \|\mathbf{w}_{l,j}\|_1 \leq \alpha_l$; similarly, $\|\mathbf{U}_l\|_{1,\infty} = \max_j \|\mathbf{u}_{l,j}\|_1 \leq \beta_l$, where $\mathbf{u}_{l,j}$ is the jth row of the matrix $\mathbf{U}_l$. As shown in (8), we can write the reweighted-RNN model recursively as $\mathbf{h}_t^{(1)} = f^{(1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{t-1}^{(d)}, \mathbf{x}_t) = \varPhi(\mathbf{W}_1\mathbf{h}_{t-1}^{(d)} + \mathbf{U}_1\mathbf{x}_t)$ and $\mathbf{h}_t^{(l)} = f^{(l)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{t-1}^{(d)}, \mathbf{x}_t) = \varPhi(\mathbf{W}_l f^{(l-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{t-1}^{(d)}, \mathbf{x}_t) + \mathbf{U}_l\mathbf{x}_t)$, where $\varPhi(\cdot)$ is an activation function. For convenience, we denote the input layer as $f^{(0)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}} = \mathbf{h}_{t-1}^{(d)}$; namely, at $t = 1$, we have $\mathbf{h}_0^{(l)} \equiv \mathbf{h}_0$.

We denote the true and training loss by $L_{\mathcal{D}}(f)$ and $L_S(f)$, respectively, where S is the training set (of size m) drawn i.i.d. from the distribution $\mathcal{D}$. The generalization error is $L_{\mathcal{D}}(f) - L_S(f)$, with f a function from the functional class $\mathcal{F}_{d,T}$. At time step t, we define $\mathbf{X}_t \in \mathbb{R}^{n\times m}$ as a matrix composed of m columns from the input vectors $\{\mathbf{x}_{t,i}\}_{i=1}^m$. We also define $\|\mathbf{X}_t\|_{2,\infty} = \sqrt{\max_{k\in\{1,\ldots,n\}} \sum_{i=1}^m \mathrm{x}_{t,i,k}^2}$ as the maximum of the ℓ_2-norms of the rows of matrix $\mathbf{X}_t$, and $\|\mathbf{h}_0\|_\infty = \max_j |\mathrm{h}_{0,j}|$.

Generalization error bound. We first derive the generalization error bound for the proposed reweighted-RNN (with T time steps) based on Rademacher complexity (see Theorem 26.5 in Shalev-Shwartz & Ben-David (2014) and Theorem C.1 in the Appendix).

Theorem 3.1 (Generalization error bound)**.** *Let $\mathcal{F}_{d,T} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}^h$ denote the functional class of reweighted-RNN with T time steps, where $\|\mathbf{W}_l\|_{1,\infty} \leq \alpha_l$, $\|\mathbf{U}_l\|_{1,\infty} \leq \beta_l$, and $1 \leq l \leq d$. Assume that the input data $\|\mathbf{X}_t\|_{2,\infty} \leq \sqrt{m}B_{\mathbf{x}}$, the initial hidden state is $\mathbf{h}_0$, and the loss function is 1-Lipschitz and bounded by η. Then, for $f \in \mathcal{F}_{d,T}$ and any $\delta > 0$, with probability at least $1-\delta$ over a training set S of size m,*

$$L_{\mathcal{D}}(f) - L_S(f) \leq 2\mathfrak{R}_S(\mathcal{F}_{d,T}) + 4\eta\sqrt{\frac{2\log(4/\delta)}{m}}, \tag{13}$$

where

$$\mathfrak{R}_S(\mathcal{F}_{d,T}) \leq \sqrt{\frac{2(4dT\log 2 + \log n + \log h)}{m}} \cdot \sqrt{\Big(\sum_{l=1}^d \beta_l \varLambda_l\Big)^2 \Big(\frac{\Lambda_0^T - 1}{\Lambda_0 - 1}\Big)^2 B_{\mathbf{x}}^2 + \varLambda_0^{2T}\|\mathbf{h}_0\|_\infty^2}, \tag{14}$$

with $\varLambda_l$ defined as follows: $\varLambda_l = \prod_{k=l+1}^d \alpha_k$ with $0 \leq l \leq d-1$ and $\varLambda_d = 1$.

Proof. The proof is given in Appendix D. □

The bound in (14) is in the order of the square root of the network depth d multiplied by the number of time steps T. The bound depends on the logarithm of the number of measurements n and the number of hidden units h. It is worth mentioning that the second square root in (14) only depends on the norm constraints and the input training data, and it is independent of the network depth d and the number of time steps T under the appropriate norm constraints.

To compare our model with ℓ_1-ℓ_1-RNN (Le et al., 2019) and Sista-RNN (Wisdom et al., 2017), we derive bounds on their Rademacher complexities for a time step t. The definitions of a functional class $\mathcal{F}_{d,t}$ for the tth time step of reweighted-RNN, ℓ_1-ℓ_1-RNN, and Sista-RNN are given in Appendix C.2. Let $\mathbf{H}_{t-1} \in \mathbb{R}^{h\times m}$ denote a matrix with columns the vectors of the previous hidden state $\{\mathbf{h}_{t-1,i}\}_{i=1}^m$, and $\|\mathbf{H}_{t-1}\|_{2,\infty} = \sqrt{\max_{k\in\{1,\ldots,h\}} \sum_{i=1}^m \mathrm{h}_{t-1,i,k}^2} \leq \sqrt{m}B_{\mathbf{h}_{t-1}}$.

Corollary 3.1.1. *The empirical Rademacher complexity of $\mathcal{F}_{d,t}$ for reweighted-RNN is bounded as*

$$\mathfrak{R}_S(\mathcal{F}_{d,t}) \leq \sqrt{\frac{2(4d\log 2 + \log n + \log h)}{m}} \cdot \sqrt{\Big(\sum_{l=1}^d \beta_l \varLambda_l\Big)^2 B_{\mathbf{x}}^2 + \varLambda_0^2 B_{\mathbf{h}_{t-1}}^2}, \tag{15}$$

with m the number of training samples and Λ_l given by $\Lambda_d = 1$, $\Lambda_l = \prod_{k=l+1}^{d} \alpha_k$ with $0 \leq l \leq d-1$.

Proof. The proof is a special case of Theorem 3.1 for time step t. □

Following the proof of Theorem 3.1, we can obtain the Rademacher complexities for ℓ_1-ℓ_1-RNN and Sista-RNN:

Corollary 3.1.2. *The empirical Rademacher complexity of $\mathcal{F}_{d,t}$ for ℓ_1-ℓ_1-RNN is bounded as:*

$$\mathfrak{R}_S(\mathcal{F}_{d,t}) \leq \sqrt{\frac{2(4d\log 2 + \log n + \log h)}{m}} \cdot \sqrt{\beta_1^2 \Big(\frac{\alpha_2^d - 1}{\alpha_2 - 1}\Big)^2 B_{\mathbf{x}}^2 + \alpha_1^2 \alpha_2^{2(d-1)} B_{\mathbf{h}_{t-1}}^2}. \tag{16}$$

Corollary 3.1.3. *The empirical Rademacher complexity of $\mathcal{F}_{d,t}$ for Sista-RNN is bounded as:*

$$\mathfrak{R}_S(\mathcal{F}_{d,t}) \leq \sqrt{\frac{2(4d\log 2 + \log n + \log h)}{m}} \cdot \sqrt{\beta_1^2 \Big(\frac{\alpha_2^d - 1}{\alpha_2 - 1}\Big)^2 B_{\mathbf{x}}^2 + \left(\alpha_1 \alpha_2^{(d-1)} + \beta_2 \Big(\frac{\alpha_2^{d-1} - 1}{\alpha_2 - 1}\Big)\right)^2 B_{\mathbf{h}_{t-1}}^2}. \tag{17}$$

By contrasting (15) with (16) and (17), we see that the complexities of ℓ_1-ℓ_1-RNN and Sista-RNN have a polynomial dependence on α_1, β_1 and α_2, β_2 (the norms of first two layers), whereas the complexity of reweighted-RNN has a polynomial dependence on $\alpha_1, \ldots, \alpha_d$ and $\beta_1, \ldots, \beta_d$ (the norms of all layers). This over-parameterization offers a flexible way to control the generalization error of reweighted-RNN. We derive empirical generalization errors in Fig. 6 in Appendix A demonstrating that increasing the depth of reweighted-RNN still ensures the low generalization error.

4 EXPERIMENTAL RESULTS

We assess the proposed RNN model in the task of video-frame reconstruction from compressive measurements. The performance is measured using the peak signal-to-noise ratio (PSNR) between the reconstructed $\hat{\mathbf{s}}_t$ and the original frame $\mathbf{s}_t$. We use the moving MNIST dataset (Srivastava et al., 2015), which contains 10K video sequences of equal length (20 frames per sequence). Similar to the setup in Le et al. (2019), the dataset is split into training, validation, and test sets of 8K, 1K, and 1K sequences, respectively. In order to reduce the training time and memory requirements, we downscale the frames from 64×64 to 16×16 pixels using bilinear decimation. After vectorizing, we obtain sequences of $\mathbf{s}_1, \ldots, \mathbf{s}_T \in \mathbb{R}^{256}$. Per sequence, we obtain measurements $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^n$ using a trainable linear sensing matrix $\mathbf{A} \in \mathbb{R}^{n \times n_0}$, with $T = 20$, $n_0 = 256$ and $n < n_0$.

We compare the reconstruction performance of the proposed reweighted-RNN model against deep-unfolding RNN models, namely, ℓ_1-ℓ_1-RNN (Le et al., 2019), Sista-RNN (Wisdom et al., 2017), and stacked-RNN models, that is, sRNN (Elman, 1990), LSTM (Hochreiter & Schmidhuber, 1997), GRU (Cho et al., 2014), FastRNN (Kusupati et al., 2018)[1], IndRNN (Li et al., 2018) and SpectralRNN (Zhang et al., 2018). For the vanilla RNN, LSTM and GRU, the native Pytorch cell implementations were used. The unfolding-based methods were implemented in Pytorch[2], with Sista-RNN and ℓ_1-ℓ_1-RNN tested by reproducing the experiments in Wisdom et al. (2017); Le et al. (2019). For FastRNN, IndRNN, and SpectralRNN cells, we use the publically available Tensorflow implementations. While Sista-RNN, ℓ_1-ℓ_1-RNN and reweighted-RNN have their own layer-stacking schemes derived from unfolding minimization algorithms, we use the stacking rule in Pascanu et al. (2013) [see Fig 2(c)] to build deep networks for other RNN architectures.

Our default settings are: a compressed sensing (CS) rate of $n/n_0 = 0.2$, $d = 3$ hidden layers[3] with $h = 2^{10}$ hidden units per layer. In each set of experiments, we vary each of these hyper-parameters while keeping the other two fixed. For the unfolding methods, the overcomplete dictionary

[1] Kusupati et al. (2018) also proposed FastGRNN; we found that, in our application scenario, the non-gated variant (the FastRNN) consistently outperformed FastGRNN. As such, we do not include results with the latter.

[2] Our implementations are available at https://1drv.ms/u/s!ApHn770BvhH2aWay9xEhAiXydfo?e=aCX1X0.

[3] In our experiments, the 2-layer LSTM network outperforms the 3-layer one (see Table 3), the default setting for LSTM is thus using 2 layers.

$\mathbf{D} \in \mathbb{R}^{n_0 \times h}$ is initialized with the discrete cosine transform (DCT) with varying dictionary sizes of $h = \{2^7,\ 2^8,\ 2^9,\ 2^{10},\ 2^{11},\ 2^{12}\}$ (corresponding to a number of hidden neurons in the other methods). For initializing λ_1, λ_2 [see (2), (3)], we perform a random search in the range of $[10^{-5}, 3.0]$ in the validation set. To avoid the problem of exploding gradients, we clip the gradients during backpropagation such that the ℓ_2-norms are less than or equal to 0.25. We do not apply weight decay regularization as we found it often leads to worse performance, especially since gradient clipping is already used for training stability. We train the networks for 200 epochs using the Adam optimizer with an initial learning rate of 0.0003, and a batch size of 32. During training, if the validation loss does not decrease for 5 epochs, we reduce the learning rate to 0.3 of its current value.

Table 1 summarizes the reconstruction results for different CS rates n/n_0. The reweighted-RNN model systematically outperforms the other models, often by a large margin. Table 2 shows similar improvements for various dimensions of hidden units. Table 3 shows that IndRNN delivers higher reconstruction performance than our model when a small number of hidden layers ($d = 1, 2$) is used. Moreover, when the depth increases, reweighted-RNN surpasses all other models.

Table 1: Average PSNR [dB] on the test set with different CS rates.

CS Rate	sRNN	LSTM	GRU	IndRNN	FastRNN	SpectralRNN	Sista-RNN	ℓ_1-ℓ_1-RNN	Ours
0.1	25.11	24.58	25.18	25.68	25.21	25.15	25.16	24.68	**26.25**
0.2	31.14	29.46	31.19	32.90	32.05	31.65	31.53	30.79	**34.19**
0.3	35.38	32.91	36.49	37.12	36.40	36.89	36.96	37.77	**42.39**
0.4	38.05	34.95	39.47	40.84	39.21	40.22	39.57	40.35	**46.03**
0.5	39.34	36.28	41.12	45.49	41.87	41.36	41.56	43.35	**48.70**

Table 2: Average PSNR [dB] on the test set with different network widths h ($\frac{n}{n_0} = 0.2$, $d = 3$).

h	sRNN	LSTM	GRU	IndRNN	FastRNN	SpectralRNN	Sista-RNN	ℓ_1-ℓ_1-RNN	Ours
$\mathbf{2^7}$	23.35	22.87	23.55	23.82	23.83	22.92	23.86	23.90	**28.09**
$\mathbf{2^8}$	25.81	23.88	26.67	27.10	26.71	24.46	29.64	29.55	**31.46**
$\mathbf{2^9}$	28.72	26.83	30.29	32.03	29.92	30.23	31.30	30.61	**33.61**
$\mathbf{2^{10}}$	31.14	29.46	31.19	32.90	32.05	31.65	31.53	30.79	**34.19**
$\mathbf{2^{11}}$	29.91	29.30	31.15	33.10	30.80	31.68	31.82	30.45	**34.80**
$\mathbf{2^{12}}$	29.71	29.08	30.93	32.47	24.26	29.26	31.63	30.09	**34.98**

Table 3: Average PSNR [dB] on the test set with different network depths d ($\frac{n}{n_0} = 0.2$, $h = 2^{10}$).

d	sRNN	LSTM	GRU	IndRNN	FastRNN	SpectralRNN	Sista-RNN	ℓ_1-ℓ_1-RNN	Ours
1	27.52	27.76	27.61	**30.12**	29.32	29.62	28.41	28.49	29.19
2	29.21	29.46	29.68	**32.73**	30.84	31.37	30.67	30.19	32.12
3	31.14	22.29	31.19	32.90	32.05	31.65	31.53	30.79	**34.19**
4	31.64	16.50	29.26	20.65	31.07	31.10	32.56	31.80	**35.99**
5	16.50	26.66	16.50	25.17	20.10	30.52	33.07	32.50	**36.91**
6	22.28	16.50	16.50	20.90	19.37	29.56	31.99	32.00	**38.90**

5 CONCLUSIONS

We designed a novel deep RNN by unfolding an algorithm that solves a reweighted ℓ_1-ℓ_1 minimization problem. Our model has high network expressivity due to *per-unit learnable activation functions* and *over-parameterized weights*. We also established the generalization error bound for the proposed model via Rademacher complexity. We showed that reweighted-RNN has good generalization properties and its error bound is tighter than existing RNNs in function of the number of time steps. Experimentation on the task of sequential video-frame reconstruction shows that our model (*i*) outperforms various state-of-the-art RNNs in terms of accuracy and (*ii*) is capable of stacking many hidden layers resulting in a better learning capability than the existing unfolding methods.

REFERENCES

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems 30*, pp. 6240–6249. 2017.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

M. Borgerding, P. Schniter, and S. Rangan. AMP-inspired deep networks for sparse linear inverse problems. *IEEE Transactions on Signal Processing*, 65(16):4293–4308, 2017.

Emmanuel J. Candès, Michael B. Wakin, and Stephen P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical linear convergence of unfolded ista and its practical weights and thresholds. In *Advances in Neural Information Processing Systems 31*, 2018.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. AdaNet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 06–11 Aug .

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11): 1413–1457, 2004.

D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

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

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Proceedings of the 31st Conference On Learning Theory*, volume 75, pp. 297–299, 06–09 Jul 2018.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 399–406, 2010.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems 31*, 2018.

Hung Duy Le, Huynh Van Luong, and Nikos Deligiannis. Designing recurrent neural networks by unfolding an l1-l1 minimization algorithm. In *Proceedings of IEEE International Conference on Image Processing*, 2019.

S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5457–5466, June 2018.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations*, 2019.

A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos. Using deep neural networks for inverse problems in imaging: beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

W. Luo, W. Liu, and S. Gao. A revisit of sparse coding based anomaly detection in stacked rnn framework. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 341–349, Oct 2017.

H. V. Luong, N. Deligiannis, J. Seiler, S. Forchhammer, and A. Kaup. Compressive online robust principal component analysis via n-ℓ_1 minimization. *IEEE Transactions on Image Processing*, 27 (9):4314–4329, Sept 2018.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning, Second edition*. MIT Press, Cambridge, Massachusetts, USA, 2018. ISBN 9780262039406.

J. F. C. Mota, N. Deligiannis, A. C. Sankaranarayanan, V. Cevher, and M. R. D. Rodrigues. Adaptive-rate reconstruction of time-varying signals with application in compressive foreground extraction. *IEEE Transactions on Signal Processing*, 64(14):3651–3666, July 2016.

J. F. C. Mota, L. Weizman, N. Deligiannis, Y. C. Eldar, and M. R. D. Rodrigues. Reference-based compressed sensing: A sample complexity approach. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

J. F. C. Mota, N. Deligiannis, and M. R. D. Rodrigues. Compressed sensing with prior information: Strategies, geometry, and bounds. *IEEE Transactions on Information Theory*, 63(7):4472–4496, July 2017.

Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1336–1343. IEEE, 2015.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Proceedings of The 28th Conference on Learning Theory*, 2015.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *Int. Conf. on Learning Representations, ICLR 2019*, 2019.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107057132, 9781107057135.

P. Sprechmann, A. M. Bronstein, and G. Sapiro. Learning efficient sparse and low rank models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1821–1833, Sep. 2015.

N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

J. Sun, H. Li, and Z. Xu. Deep ADMM-Net for compressive sensing MRI. In *Advances in Neural Information Processing Systems*, pp. 10–18, 2016.

S. Wisdom, T. Powers, J. Pitton, and L. Atlas. Building recurrent networks by unfolding iterative thresholding for sequential sparse recovery. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017.

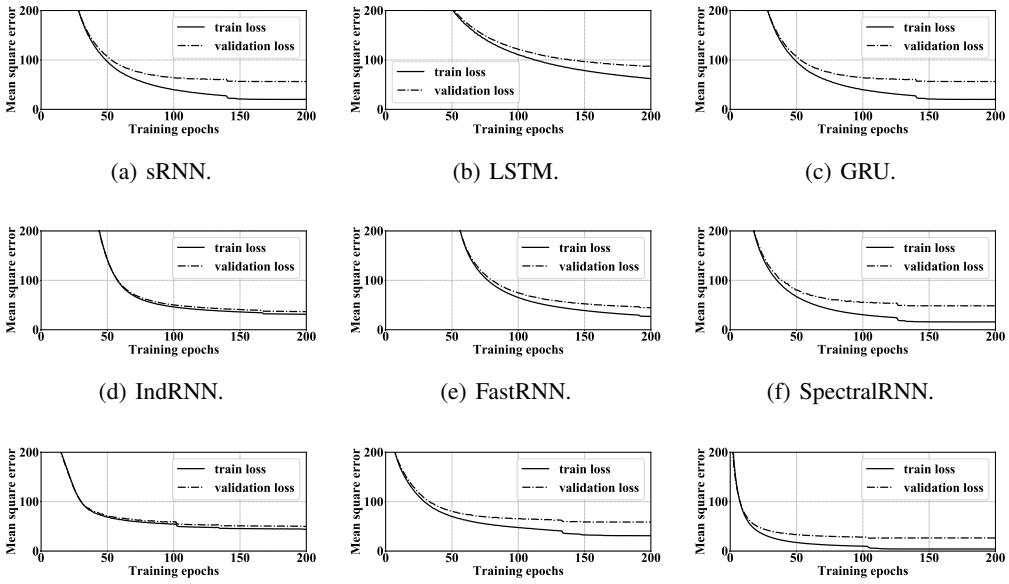
B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang. Maximal sparsity with deep networks? In *Advances in Neural Information Processing Systems*, pp. 4340–4348, 2016.

Jiong Zhang, Qi Lei, and Inderjit S. Dhillon. Stabilizing gradients for deep neural networks via efficient SVD parameterization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018.

A SUPPLEMENTARY EXPERIMENTS

In our experiments, we use the publically available Tensorflow implementations for FastRNN[4], IndRNN[5], and SpectralRNN[6] cells. While Sista-RNN, ℓ_1-ℓ_1-RNN and reweighted-RNN have their own layer-stacking schemes derived from unfolding minimization algorithms, we use the stacking rule in Pascanu et al. (2013) [see Fig 2(c)] to build deep networks for other RNN models.

Figure 3 shows the learning curves of all methods under the default setting. It can be seen that reweighted-RNN achieves the lowest mean square error on both the training and validation sets. It can also be observed that the unfolding methods converge faster than the stacked RNNs, with the proposed reweighted-RNN being the fastest. More experimental results for the proposed reweighted-RNN are provided to illustrate the learning curves, which measure the average mean square error vs. the training epochs between the original frames and their reconstructed counterparts, with different CS rates [Fig. 4], different network depths d [Fig. 6], and different network widths h [Fig. 5].



(a) sRNN. (b) LSTM. (c) GRU.

(d) IndRNN. (e) FastRNN. (f) SpectralRNN.

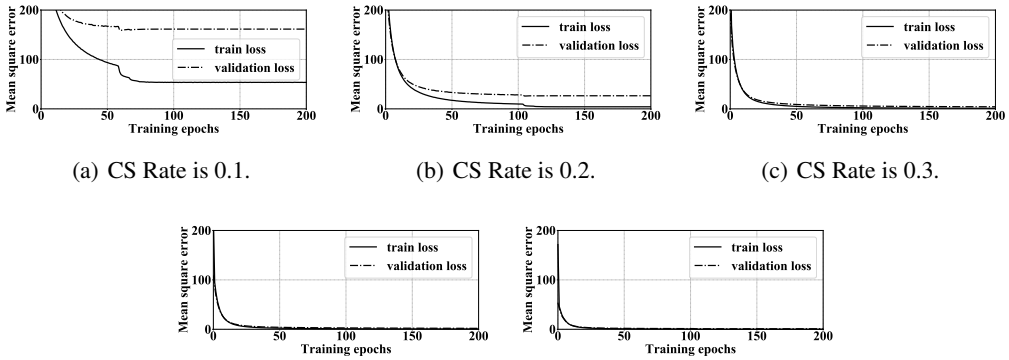
(g) Sista-RNN. (h) ℓ_1-ℓ_1-RNN. (i) reweighted-RNN.

Figure 3: Average mean square error between the original and reconstructed frames vs. training epoches on the training and the validation sets for the default setting (a CS rate is 0.2, $d = 3$, $h = 2^{10}$).

[4]Code available at https://github.com/microsoft/EdgeML

[5]Code available at https://github.com/batzner/indrnn

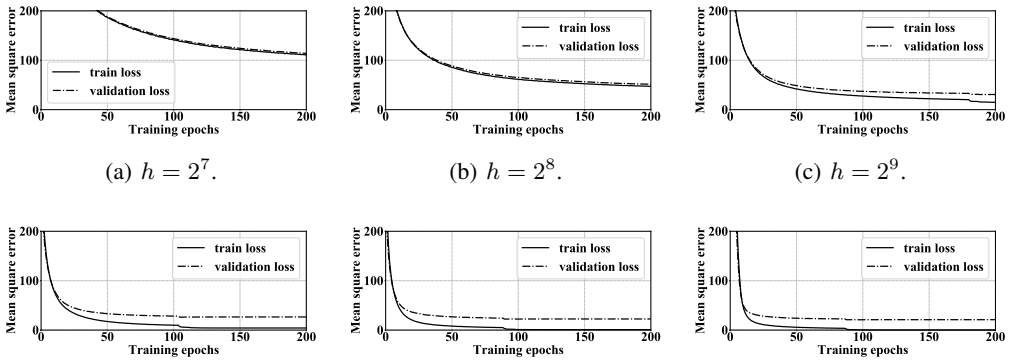
[6]Code available at https://github.com/zhangjiong724/spectral-RNN



(a) CS Rate is 0.1. (b) CS Rate is 0.2. (c) CS Rate is 0.3.

(d) CS Rate is 0.4. (e) CS Rate is 0.5.

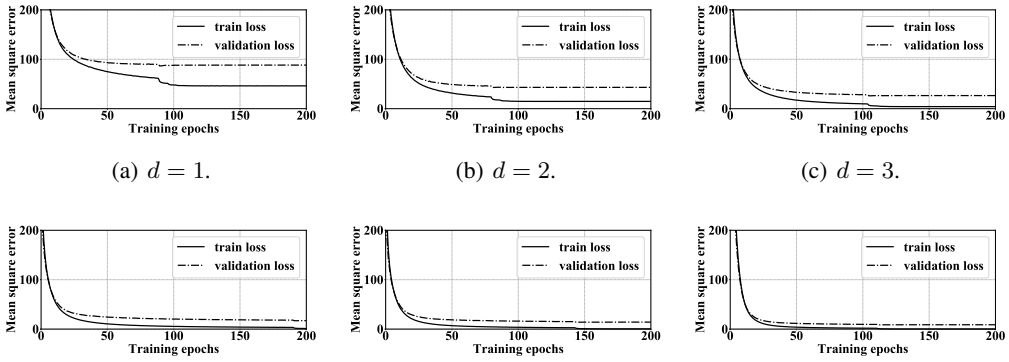
Figure 4: Reweighted-RNN: Average mean square error between the original and reconstructed frames vs. training epoches on the training and the validation sets with different CS rates ($d = 3$, $h = 2^{10}$).



(a) $h = 2^7$. (b) $h = 2^8$. (c) $h = 2^9$.

(d) $h = 2^{10}$. (e) $h = 2^{11}$. (f) $h = 2^{12}$.

Figure 5: Reweighted-RNN: Average mean square error between the original and reconstructed frames vs. training epoches on the training and the validation sets with different network widths h (a CS rates is 0.2, $d = 3$).



(a) $d = 1$. (b) $d = 2$. (c) $d = 3$.

(d) $d = 4$. (e) $d = 5$. (f) $d = 6$.

Figure 6: Reweighted-RNN: Average mean square error between the original and reconstructed frames vs. training epoches on the training and the validation sets with different network depths d (a CS rate is 0.2, $h = 2^{10}$).

B THE PROXIMAL OPERATOR FOR PROBLEM (3)

Proposition B.1. *The proximal operator $\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar}(u)$ in (5) for the reweighted ℓ_1-ℓ_1 minimization problem (3), for which $g(v) = \lambda_1\mathsf{g}|v| + \lambda_2\mathsf{g}|v-\hbar|$, is given by*

$$\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar\geq 0}(u) = \begin{cases} u - \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}, & \hbar + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} < u < \infty \\ \hbar, & \hbar + \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \hbar + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} \\ u - \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} < u < \hbar + \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \\ 0, & -\frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \\ u + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & -\infty < u < -\frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \end{cases} \tag{18}$$

$$\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar<0}(u) = \begin{cases} u - \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}, & \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} < u < \infty \\ 0, & -\frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} \\ u + \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}, & \hbar - \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} < u < -\frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} \\ \hbar, & \hbar - \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \hbar - \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} \\ u - \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & -\infty < u < \hbar - \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \end{cases} \tag{19}$$

Proof. We compute the proximal operator $\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar}(u)$ (18) for $\hbar \geq 0$, it is similar for $\hbar < 0$. From (5), $\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar}(u)$ is expressed by:

$$\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar}(u) = \arg\min_{v\in\mathbb{R}} \Big\{\varphi(v) := \frac{\lambda_1}{c}\mathsf{g}|v| + \frac{\lambda_2}{c}\mathsf{g}|v-\hbar| + \frac{1}{2}|v-u|^2\Big\}. \tag{20}$$

We first consider the $\partial\varphi(v)/\partial v$ in $v \in \{(-\infty, 0), (0, \hbar), (\hbar, \infty)\}$, in which $\partial\varphi(v)$ exists. Taking the derivative of $\varphi(v)$ in these intervals delivers

$$\frac{\partial\varphi(v)}{\partial v} = \frac{\lambda_1}{c}\mathsf{g}\cdot\mathrm{sign}(v) + \frac{\lambda_2}{c}\mathsf{g}\cdot\mathrm{sign}(v-\hbar) + (v-u), \tag{21}$$

where $\mathrm{sign}(.)$ is a sign function. When setting $\partial\varphi(v)/\partial v = 0$ to minimize $\varphi(v)$, we derive:

$$v = \begin{cases} u - \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}, & \hbar < v < \infty \\ u - \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & 0 < v < \hbar \\ u + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & -\infty < v < 0 \end{cases} \tag{22}$$

From (20) and (22), we have

$$\Phi_{\frac{\lambda_1}{c}\mathsf{g},\frac{\lambda_2}{c}\mathsf{g},\hbar}(u) = \begin{cases} u - \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}, & \hbar + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} < u < \infty \\ u - \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} < u < \hbar + \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \\ u + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}, & -\infty < u < -\frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \end{cases} \tag{23}$$

In the remaining range value of u, namely, $-\frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}$ and $\hbar + \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \hbar + \frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g}$, we prove that the minimum of $\varphi(v)$ (20) is obtained when $v = 0$ and $v = \hbar$, respectively.

Let us rewrite $\varphi(v)$, which was defined in (20), as

$$\varphi(v) = \frac{\lambda_1}{c}\mathsf{g}|v| + \frac{\lambda_2}{c}\mathsf{g}|v-\hbar| + \frac{1}{2}|v-u|^2 \tag{24}$$

By applying the inequality $|a-b| \geq |a| - |b|$, where $a, b \in \mathbb{R}$, to (24), we obtain:

$$\begin{aligned} \varphi(v) &\geq \frac{\lambda_1}{c}\mathsf{g}|v| + \frac{\lambda_2}{c}\mathsf{g}|v| - \frac{\lambda_2}{c}\mathsf{g}|\hbar| + \frac{1}{2}v^2 - vu + \frac{1}{2}u^2 \\ &\geq |v|\Big(\frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} - |u|\Big) + \frac{1}{2}v^2 - \frac{\lambda_2}{c}\mathsf{g}|\hbar| + \frac{1}{2}u^2 \end{aligned} \tag{25}$$

For $-\frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g} \leq u \leq \frac{\lambda_1}{c}\mathsf{g} - \frac{\lambda_2}{c}\mathsf{g}$, from (25), $\varphi(v)$ is minimal when $v = 0$, due to $\frac{\lambda_1}{c}\mathsf{g} + \frac{\lambda_2}{c}\mathsf{g} - |u| \geq 0$.

Similarly, for $\hbar+\frac{\lambda_1}{c}\mathsf{g}-\frac{\lambda_2}{c}\mathsf{g}\leq u\leq\hbar+\frac{\lambda_1}{c}\mathsf{g}+\frac{\lambda_2}{c}\mathsf{g}$, i.e., $\frac{\lambda_1}{c}\mathsf{g}-\frac{\lambda_2}{c}\mathsf{g}\leq u-\hbar\leq\frac{\lambda_1}{c}\mathsf{g}+\frac{\lambda_2}{c}\mathsf{g}$, we have

$$\begin{aligned}\varphi(v)\geq&\frac{\lambda_1}{c}\mathsf{g}|v-\hbar|-\frac{\lambda_1}{c}\mathsf{g}|\hbar|+\frac{\lambda_2}{c}\mathsf{g}|v-\hbar|+\frac{1}{2}(v-\hbar)^2-|v-\hbar||u-\hbar|+\frac{1}{2}(u-\hbar)^2\\ \geq&|v-\hbar|\Big(\frac{\lambda_1}{c}\mathsf{g}+\frac{\lambda_2}{c}\mathsf{g}-|u-\hbar|\Big)+\frac{1}{2}(v-\hbar)^2-\frac{\lambda_1}{c}\mathsf{g}|\hbar|+\frac{1}{2}(u-\hbar)^2.\end{aligned}\tag{26}$$

From (26), $\varphi(v)$ is minimal when $v=\hbar$, since $\frac{\lambda_1}{c}\mathsf{g}+\frac{\lambda_2}{c}\mathsf{g}-|u-\hbar|\geq 0$. Combining these results with the result in (23), we conclude the proof. □

C Generalization and deep unfolding RNNs

C.1 Generalization error bound definition

Notations. Let $f^{(d)}_{\boldsymbol{\mathcal{W}}}:\mathbb{R}^n\mapsto\mathbb{R}^h$ be the function computed by a d-layer network with weight parameters $\boldsymbol{\mathcal{W}}$. The network $f^{(d)}_{\boldsymbol{\mathcal{W}}}$ maps an input sample $\mathbf{x}_i\in\mathbb{R}^n$ (from an input space $\mathcal{X}$) to an output $\mathbf{y}_i\in\mathbb{R}^h$ (from an output space $\mathcal{Y}$), i.e., $\mathbf{y}_i=f^{(d)}_{\boldsymbol{\mathcal{W}}}(\mathbf{x}_i)$. Let S denote a training set of size m, i.e., $S=\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^m$ and $\mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim S}[\cdot]$ denote an expectation over $(\mathbf{x}_i,\mathbf{y}_i)$ from S. The set S is drawn i.i.d. from a distribution $\mathcal{D}$, denoted as $S\sim\mathcal{D}^m$, over a space $\mathcal{Z}=\mathcal{X}\times\mathcal{Y}$. Let $\mathcal{F}$ be a (class) set of functions. Let $\ell:\mathcal{F}\times\mathcal{Z}\mapsto\mathbb{R}$ denote the loss function and $\ell\circ\mathcal{F}=\{z\mapsto\ell(f,z):f\in\mathcal{F}\}$. We define the true loss and the empirical (training) loss by $L_{\mathcal{D}}(f)$ and $L_S(f)$, respectively, as follows:

$$L_{\mathcal{D}}(f)=\mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim\mathcal{D}}\big[\ell\big(f(\mathbf{x}_i),\mathbf{y}_i\big)\big],\tag{27}$$

and

$$L_S(f)=\mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim S}\big[\ell\big(f(\mathbf{x}_i),\mathbf{y}_i\big)\big].\tag{28}$$

The *generalization error*, which is defined as a measure of how accurately a learned algorithm is able to predict outcome values for unseen data, is calculated by $L_{\mathcal{D}}(f)-L_S(f)$.

Rademacher complexity. Let $\mathcal{F}$ be a hypothesis set of functions (neural networks). The *empirical Rademacher complexity* of $\mathcal{F}$ (Shalev-Shwartz & Ben-David, 2014) for a training sample set S is defined as follows:

$$\mathfrak{R}_S(\mathcal{F})=\frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\sup_{f\in\mathcal{F}}\sum_{i=1}^m\epsilon_i f(\mathbf{x}_i)\right],\tag{29}$$

where $\boldsymbol{\epsilon}=(\epsilon_1,...,\epsilon_m)$; here ϵ_i are independent uniformly distributed random (Rademacher) variables from $\{\pm1\}$, according to $\mathbb{P}[\epsilon_i=1]=\mathbb{P}[\epsilon_i=-1]=1/2$.

The generalization error bound (Shalev-Shwartz & Ben-David, 2014) is derived based on the Rademacher complexity defined in the following theorem:

Theorem C.1. *(Shalev-Shwartz & Ben-David, 2014, Theorem 26.5)*
Assume that $|\ell(f,z)|\leq\eta$ for all $f\in\mathcal{F}$ and z. Then, for any $\delta>0$, with probability at least $1-\delta$,

$$L_{\mathcal{D}}(f)-L_S(f)\leq 2\mathfrak{R}_S(\ell\circ\mathcal{F})+4\eta\sqrt{\frac{2\log(4/\delta)}{m}}.\tag{30}$$

It can be noted that the bound in (30) depends on the training set S, which makes it applicable to a number of learning problems, e.g., regression and classification, under given a loss function ℓ.

C.2 Notation for Deep Unfolded RNNs

In this subsection, we provide the required notation for the proposed reweighted-RNN model, ℓ_1-ℓ_1-RNN, and Sista-RNN, which will be used in the derivation of their respective generalization analysis.

The proposed reweighted-RNN. Let $\mathbf{h}^{(l)}_t$ be the hidden states in layer l evolving in time step t. We write the reweighted-RNN model recursively as $\mathbf{h}^{(1)}_t=f^{(1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{t-1},\mathbf{x}_t)=\varPhi(\mathbf{W}_1\mathbf{h}^{(d)}_{t-1}+\mathbf{U}_1\mathbf{x}_t)$ and $\mathbf{h}^{(l)}_t=f^{(l)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{t-1},\mathbf{x}_t)=\varPhi\Big(\mathbf{W}_l f^{(l-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{t-1},\mathbf{x}_t)+\mathbf{U}_l\mathbf{x}_t\Big)$, where $\varPhi$ is an activation function. The

hidden state is updated as shown in (8). The real-valued family of functions, $\mathcal{F}_{d,t} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$, for the functions $f^{(d)}_{\boldsymbol{\mathcal{W}},\mathbf{U}}$ in layer d is defined by:

$$\mathcal{F}_{d,t} = \Big\{ (\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) \mapsto \varPhi(\mathbf{w}_d^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) + \mathbf{u}_d^{\mathrm{T}} \mathbf{x}_t) : \|\mathbf{W}_d\|_{1,\infty} \leq \alpha_d,\ \|\mathbf{U}_d\|_{1,\infty} \leq \beta_d \Big\}, \tag{31}$$

where α_l, β_l are nonnegative hyper-parameters for layer l, where $1 < l \leq d$. In layer $l = 1$, the real-valued family of functions, $\mathcal{F}_{1,t} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$, for the functions $f^{(1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}$ is defined by:

$$\mathcal{F}_{1,t} = \Big\{ (\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) \mapsto \varPhi(\mathbf{w}_1^{\mathrm{T}} \mathbf{h}^{(d)}_{t-1} + \mathbf{u}_1^{\mathrm{T}} \mathbf{x}_t) : \|\mathbf{W}_1\|_{1,\infty} \leq \alpha_1,\ \|\mathbf{U}\|_{1,\infty} \leq \beta_1 \Big\}, \tag{32}$$

where α_1, β_1 are nonnegative hyper-parameters. We denote the input layer as $f^{(0)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}} = \mathbf{h}^{(d)}_{t-1}$, in particular, at $t = 1$, $\mathbf{h}^{(l)}_0 \equiv \mathbf{h}_0$.

The ℓ_1-ℓ_1-RNN model (Le et al., 2019). The hidden state $\mathbf{h}^{(l)}_t$ for ℓ_1-ℓ_1-RNN is updated as shown in (12). The real-valued family of functions, $\mathcal{F}_{d,t} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$, for the function $f^{(d)}_{\boldsymbol{\mathcal{W}},\mathbf{U}}$ in layer d is defined by:

$$\mathcal{F}_{d,t} = \Big\{ (\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) \mapsto \varPhi(\mathbf{w}_2^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) + \mathbf{u}_1^{\mathrm{T}} \mathbf{x}_t) : \|\mathbf{W}_2\|_{1,\infty} \leq \alpha_2,\ \|\mathbf{U}_1\|_{1,\infty} \leq \beta_1 \Big\}, \tag{33}$$

where α_2, β_1 are nonnegative hyper-parameters for layer l, where $1 < l \leq d$. In layer $l = 1$, the real-valued family of functions, $\mathcal{F}_{1,t} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$, for the functions $f^{(1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}$ is defined by:

$$\mathcal{F}_{1,t} = \Big\{ (\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) \mapsto \varPhi(\mathbf{w}_1^{\mathrm{T}} \mathbf{h}^{(d)}_{t-1} + \mathbf{u}_1^{\mathrm{T}} \mathbf{x}_t) : \|\mathbf{W}_1\|_{1,\infty} \leq \alpha_1,\ \|\mathbf{U}\|_{1,\infty} \leq \beta_1 \Big\}, \tag{34}$$

where α_1, β_1 are nonnegative hyper-parameters.

The Sista-RNN model (Wisdom et al., 2017). The hidden state $\mathbf{h}^{(l)}_t$ in Sista-RNN is updated by:

$$\mathbf{h}^{(l)}_t = \begin{cases} \phi\Big(\mathbf{W}_1 \mathbf{h}^{(d)}_{t-1} + \mathbf{U}_1 \mathbf{x}_t\Big), & l = 1, \\ \phi\Big(\mathbf{W}_2 \mathbf{h}^{(l-1)}_t + \mathbf{U}_1 \mathbf{x}_t + \mathbf{U}_2 \mathbf{h}^{(d)}_{t-1}\Big), & l > 1, \end{cases} \tag{35}$$

The real-valued family of functions, $\mathcal{F}_{d,t} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$, for the functions $f^{(d)}_{\boldsymbol{\mathcal{W}},\mathbf{U}}$ in layer d is defined by:

$$\begin{aligned} \mathcal{F}_{d,t} = \Big\{ & (\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) \mapsto \phi\Big(\mathbf{w}_2^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) + \mathbf{u}_1^{\mathrm{T}} \mathbf{x}_t + \mathbf{u}_2^{\mathrm{T}} \mathbf{h}^{(d)}_{t-1}\Big) \\ & : \|\mathbf{W}_2\|_{1,\infty} \leq \alpha_2, \|\mathbf{U}\|_{1,\infty} \leq \beta_1,\ \|\mathbf{U}\|_{2,\infty} \leq \beta_2 \Big\}, \end{aligned} \tag{36}$$

where $\alpha_2, \beta_1, \beta_2$ are nonnegative hyper-parameters. In layer $l = 1$,

$$\mathcal{F}_{1,t} = \Big\{ (\mathbf{h}^{(d)}_{t-1}, \mathbf{x}_t) \mapsto \phi\Big(\mathbf{w}_1^{\mathrm{T}} \mathbf{h}^{(d)}_{t-1} + \mathbf{u}_1^{\mathrm{T}} \mathbf{x}_t\Big) : \|\mathbf{W}_1\|_{1,\infty} \leq \alpha_1,\ \|\mathbf{U}\|_{1,\infty} \leq \beta_1 \Big\}, \tag{37}$$

where α_1, β_1 are nonnegative hyper-parameters.

D Proof of Theorem 3.1

Proof. We consider the real-valued family of functions $\mathcal{F}_{d,T} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$ for the functions $f^{(d)}_{\boldsymbol{\mathcal{W}},\mathbf{U}}$ to update $\mathbf{h}^{(d)}_T$ in layer d, time step T, defined as

$$\mathcal{F}_{d,T} = \Big\{ (\mathbf{h}^{(d)}_{T-1}, \mathbf{x}_T) \mapsto \varPhi(\mathbf{w}_d^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}^{(d)}_{T-1}, \mathbf{x}_T) + \mathbf{u}_d^{\mathrm{T}} \mathbf{x}_T) : \|\mathbf{W}_d\|_{1,\infty} \leq \alpha_d,\ \|\mathbf{U}_d\|_{1,\infty} \leq \beta_d \Big\}, \tag{38}$$

where $\mathbf{w}_d, \mathbf{u}_d$ are the corresponding rows from $\mathbf{W}_d, \mathbf{U}_d$, respectively, and α_l, β_l, with $1 < l \leq d$, are nonnegative hyper-parameters. For the first layer and the first time step, i.e., $l = 1$, $t = 1$, the real-valued family of functions, $\mathcal{F}_{1,1} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}$, for the functions $f^{(1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}$ is defined by:

$$\mathcal{F}_{1,1} = \Big\{ (\mathbf{h}_0, \mathbf{x}_1) \mapsto \varPhi(\mathbf{w}_1^{\mathrm{T}} \mathbf{h}_0 + \mathbf{u}_1^{\mathrm{T}} \mathbf{x}_1) : \|\mathbf{W}_1\|_{1,\infty} \leq \alpha_1,\ \|\mathbf{U}\|_{1,\infty} \leq \beta_1 \Big\}, \tag{39}$$

where α_1, β_1 are nonnegative hyper-parameters. We denote the input layer as $f^{(0)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}} = \mathbf{h}_0$ at the first time step. From the definition of Rademacher complexity in (29) and the family of functions in (38) and (39), we obtain:

$$m\mathfrak{R}_S(\mathcal{F}_{d,T}) \tag{40a}$$

$$\leq \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m} \Bigg[\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_d\|_1 \leq \alpha_d \\ \|\mathbf{u}_d\|_1 \leq \beta_d}} \sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_d^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{T-1,i}, \mathbf{x}_{T,i}) + \mathbf{u}_d^{\mathrm{T}} \mathbf{x}_{T,i} \Big) \Bigg]$$

$$\leq \frac{1}{\lambda} \log \exp \Bigg(\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m} \Bigg[\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_d\|_1 \leq \alpha_d \\ \|\mathbf{u}_d\|_1 \leq \beta_d}} \lambda \sum_{i=1}^m \epsilon_i \Big(\mathbf{w}_d^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{T-1,i}, \mathbf{x}_{T,i}) + \mathbf{u}_d^{\mathrm{T}} \mathbf{x}_{T,i} \Big) \Bigg] \Bigg)$$

$$\leq \frac{1}{\lambda} \log \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m} \Bigg[\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_d\|_1 \leq \alpha_d \\ \|\mathbf{u}_d\|_1 \leq \beta_d}} \exp \Bigg(\lambda \sum_{i=1}^m \epsilon_i \Big(\mathbf{w}_d^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{T-1,i}, \mathbf{x}_{T,i}) \Big) + \lambda \sum_{i=1}^m \epsilon_i \mathbf{u}_d^{\mathrm{T}} \mathbf{x}_{T,i} \Bigg) \Bigg] \tag{40b}$$

$$\leq \frac{1}{\lambda} \log \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m} \Bigg[\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_d\|_1 \leq \alpha_d}} \exp \Bigg(\lambda \sum_{i=1}^m \epsilon_i \Big(\mathbf{w}_d^{\mathrm{T}} f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{T-1,i}, \mathbf{x}_{T,i}) \Big) \Bigg)$$

$$\cdot \sup_{\|\mathbf{u}_d\|_1 \leq \beta_d} \exp \Bigg(\lambda \sum_{i=1}^m \epsilon_i \mathbf{u}_d^{\mathrm{T}} \mathbf{x}_{T,i} \Bigg) \Bigg], \tag{40c}$$

where $\lambda > 0$ is an arbitrary parameter, Eq. (40b) follows Lemma D.1 for 1-Lipschitz $\varPhi$ a long with Inequality (61), and (40c) holds by Inequality (58).

For layer $1 \leq l \leq d$ and time step t, let us denote:

$$\varDelta^{(l)}_{\mathbf{h}_{t-1},\mathbf{x}_t} = \sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_l\|_1 \leq \alpha_l}} \exp \Bigg(\lambda \varLambda_l \sum_{i=1}^m \epsilon_i \Big(\mathbf{w}_l^{\mathrm{T}} f^{(l-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{t-1,i}, \mathbf{x}_{t,i}) \Big) \Bigg), \tag{41}$$

$$\varDelta^{(l)}_{\mathbf{x}_t} = \sup_{\|\mathbf{u}_l\|_1 \leq \beta_l} \exp \Bigg(\lambda \varLambda_l \sum_{i=1}^m \epsilon_i \Big(\mathbf{u}_l^{\mathrm{T}} \mathbf{x}_{t,i} \Big) \Bigg), \tag{42}$$

where $\varLambda_l$ is defined as follows: $\varLambda_d = 1$, $\varLambda_l = \prod_{k=l+1}^{d} \alpha_k$ with $1 \leq l \leq d-1$, and $\varLambda_0 = \prod_{k=1}^{d} \alpha_k$.

Following the Hölder's inequality in (57) in case of $p = 1$ and $q = \infty$ applied to $\mathbf{w}_l^{\mathrm{T}}$ and $f^{(l-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{t-1,i}, \mathbf{x}_{t,i})$ in (41), respectively, we get:

$$\varDelta^{(d)}_{\mathbf{h}_{t-1},\mathbf{x}_t} \tag{43}$$

$$\leq \sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{W}_{d-1}\|_{1,\infty} \leq \alpha_{d-1} \\ \|\mathbf{U}_{d-1}\|_{1,\infty} \leq \beta_{d-1}}} \exp \Bigg(\lambda \alpha_d \Bigg\| \sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{W}_{d-1} f^{(d-2)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{t-1,i}, \mathbf{x}_{t,i}) + \mathbf{U}_{d-1} \mathbf{x}_{t,i} \Big) \Bigg\|_\infty \Bigg)$$

$$\leq \sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_{d-1,k}\|_1 \leq \alpha_{d-1} \\ \|\mathbf{u}_{d-1,k}\|_1 \leq \beta_{d-1}}} \exp\left(\lambda\alpha_d \max_{k\in\{1,\cdots,h\}} \left|\sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{t-1,i},\mathbf{x}_{t,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{t,i}\Big)\right|\right)$$

$$\leq \sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_{d-1,k}\|_1 \leq \alpha_{d-1} \\ \|\mathbf{u}_{d-1,k}\|_1 \leq \beta_{d-1}}} \exp\left(\lambda\alpha_d \left|\sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{t-1,i},\mathbf{x}_{t,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{t,i}\Big)\right|\right). \tag{44}$$

Similarly, from (42), we obtain:

$$\varDelta_{\mathbf{x}_t}^{(d)} \leq \sup_{\|\mathbf{u}_d\|_1 \leq \beta_d} \exp\left(\lambda \sum_{i=1}^m \epsilon_i \mathbf{u}_d^{\mathrm{T}}\mathbf{x}_{t,i}\right) \leq \exp\left(\lambda\beta_d \Big\|\sum_{i=1}^m \epsilon_i \mathbf{x}_{t,i}\Big\|_\infty\right) \leq \exp\left(\lambda\beta_d \Big|\sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\Big|\right), \tag{45}$$

where $\{\tau,\kappa\} = \underset{t\in\{1,\ldots,T\},j\in\{1,\ldots,n\}}{\operatorname{argmax}} \Big|\sum_{i=1}^m \epsilon_i \mathrm{x}_{t,i,j}\Big|$.

From (40c), (43), and (45), we get:

$$\begin{aligned}
&m\mathfrak{R}_S(\mathcal{F}_{d,T}) \\
&\leq \frac{1}{\lambda}\log\left(\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_{d-1,k}\|_1 \leq \alpha_{d-1} \\ \|\mathbf{u}_{d-1,k}\|_1 \leq \beta_{d-1}}} \exp\left(\lambda\alpha_d \left|\sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{T-1,i},\mathbf{x}_{T,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{T,i}\Big)\right|\right.\right.\right. \\
&\qquad \left.\left.\left. + \lambda\beta_d \Big|\sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\Big|\right)\right]\right) \\
&\leq \frac{1}{\lambda}\log\left(\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}} \\ \|\mathbf{w}_{d-1,k}\|_1 \leq \alpha_{d-1} \\ \|\mathbf{u}_{d-1,k}\|_1 \leq \beta_{d-1}}} \Bigg(\right.\right. \\
&\qquad \exp\left(\lambda\alpha_d \sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{T-1,i},\mathbf{x}_{T,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{T,i}\Big) + \lambda\beta_d \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right) \\
&\qquad + \exp\left(\lambda\alpha_d \sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{T-1,i},\mathbf{x}_{T,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{T,i}\Big) - \lambda\beta_d \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right) \\
&\qquad + \exp\left(-\lambda\alpha_d \sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{T-1,i},\mathbf{x}_{T,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{T,i}\Big) + \lambda\beta_d \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right) \\
&\qquad \left.\left.\left. + \exp\left(-\lambda\alpha_d \sum_{i=1}^m \epsilon_i \varPhi\Big(\mathbf{w}_{d-1,k}^{\mathrm{T}} f_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}^{(d-2)}(\mathbf{h}_{T-1,i},\mathbf{x}_{T,i}) + \mathbf{u}_{d-1,k}^{\mathrm{T}}\mathbf{x}_{T,i}\Big) - \lambda\beta_d \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right)\right)\right]\right) \\
&\leq \frac{1}{\lambda}\log\left(4 \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\varDelta_{\mathbf{h}_{T-1},\mathbf{x}_T}^{(d-1)} \varDelta_{\mathbf{x}_T}^{(d-1)} \exp\left(\beta_d\lambda \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right)\right]\right) && \text{(46a)} \\
&\leq \frac{1}{\lambda}\log\left(4^{d-1} \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\varDelta_{\mathbf{h}_{T-1},\mathbf{x}_T}^{(1)} \varDelta_{\mathbf{x}_T}^{(1)} \exp\left(\lambda\Big(\sum_{l=2}^d \beta_l \varLambda_l\Big) \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right)\right]\right) && \text{(46b)} \\
&\leq \frac{1}{\lambda}\log\left(4^{d-1} \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(\lambda\Big(\sum_{l=2}^d \beta_l \varLambda_l\Big) \sum_{i=1}^m \epsilon_i \mathrm{x}_{\tau,i,\kappa}\right) \sup_{\|\mathbf{w}_1\|_1 \leq \alpha_1} \exp\left(\lambda\varLambda_1 \sum_{i=1}^m \epsilon_i \Big(\mathbf{w}_1^{\mathrm{T}}\mathbf{h}_{T-1,i}\Big)\right)\right.\right. \\
&\qquad \left.\left. \cdot \sup_{\|\mathbf{u}_1\|_1 \leq \beta_1} \exp\left(\lambda\varLambda_1 \sum_{i=1}^m \epsilon_i \Big(\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_{T,i}\Big)\right)\right]\right) && \text{(46c)}
\end{aligned}$$

$$\leq \frac{1}{\lambda}\log\left(4^{d-1}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(\lambda\Big(\sum_{l=2}^{d}\beta_l\varLambda_l\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,\kappa}\right)\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}\\ \|\mathbf{w}_d\|_1\leq\alpha_d\\ \|\mathbf{u}_d\|_1\leq\beta_d}}\exp\left(\lambda\varLambda_0\Big\|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{T-1,i}\Big\|_\infty\right)\right.\right.$$

$$\left.\left.\cdot\exp\left(\lambda\beta_1\varLambda_1\Big\|\sum_{i=1}^{m}\epsilon_i\mathbf{x}_{T,i}\Big\|_\infty\right)\right)\right] \tag{46d}$$

$$\leq \frac{1}{\lambda}\log\left(4^{d}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,\kappa}\right)\right.\right.$$

$$\left.\left.\cdot\sup_{\substack{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}\\ \|\mathbf{w}_d\|_1\leq\alpha_d\\ \|\mathbf{u}_d\|_1\leq\beta_d}}\exp\left(\lambda\varLambda_0\sum_{i=1}^{m}\epsilon_i\varPhi\Big(\mathbf{w}_d^{\mathrm{T}}f^{(d-1)}_{\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{U}}}(\mathbf{h}_{T-2,i},\mathbf{x}_{T-1,i})+\mathbf{u}_d^{\mathrm{T}}\mathbf{x}_{T-1,i}\Big)\right)\right]\right), \tag{46e}$$

where (46a) holds by inequality (58), and (46b) follows by repeating the process from layer $d-1$ to layer 1 for time step T. Furthermore, (46c) is obtained as the beginning of the process for time step $T-1$ and (46d) follows inequality (57).

Proceeding by repeating the above procedure in (46e) from time step $T-1$ to time step 1, we get:

$$m\mathfrak{R}_S(\mathcal{F}_{d,T})$$

$$\leq \frac{1}{\lambda}\log\left(4^{dT}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,\kappa}\right)\exp\left(\lambda\varLambda_0^T\Big\|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_0\Big\|_\infty\right]\right). \tag{47}$$

Let us denote $\mu=\mathop{\mathrm{argmax}}_{j\in\{1,\dots,h\}}\Big|\sum_{i=1}^{m}\epsilon_i\mathrm{h}_{0,j}\Big|$, from (47), we have:

$$m\mathfrak{R}_S(\mathcal{F}_{d,T})$$

$$\leq \frac{1}{\lambda}\log\left(4^{dT}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,\kappa}\right)\exp\left(\lambda\varLambda_0^T\sum_{i=1}^{m}\epsilon_i\mathrm{h}_{0,\mu}\right)\right]\right)$$

$$\leq \frac{2dT\log 2}{\lambda}+\frac{1}{2\lambda}\log\left(\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,\kappa}\right)\right.\right.$$

$$\left.\left.\cdot\exp\left(\lambda\varLambda_0^T\sum_{i=1}^{m}\epsilon_i\mathrm{h}_{0,\mu}\right)\right]\right)^2$$

$$\leq \frac{2dT\log 2}{\lambda}+\frac{1}{2\lambda}\log\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(2\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,\kappa}\right)\right]$$

$$+\frac{1}{2\lambda}\log\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(2\lambda\varLambda_0^T\sum_{i=1}^{m}\epsilon_i\mathrm{h}_{0,\mu}\right)\right] \tag{48a}$$

$$\leq \frac{2dT\log 2}{\lambda}+\frac{1}{2\lambda}\log\sum_{j=1}^{n}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(2\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\sum_{i=1}^{m}\epsilon_i\mathrm{x}_{\tau,i,j}\right)\right]$$

$$+\frac{1}{2\lambda}\log\sum_{j=1}^{h}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(2\lambda\varLambda_0^T\sum_{i=1}^{m}\epsilon_i\mathrm{h}_{0,j}\right)\right] \tag{48b}$$

$$\leq \frac{2dT\log 2}{\lambda}+\frac{1}{2\lambda}\log\sum_{j=1}^{n}\prod_{i=1}^{m}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(2\lambda\Big(\sum_{l=1}^{d}\beta_l\varLambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\epsilon_i\mathrm{x}_{\tau,i,j}\right)\right]$$

$$+\frac{1}{2\lambda}\log\sum_{j=1}^{h}\prod_{i=1}^{m}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\exp\left(2\lambda\varLambda_0^T\epsilon_i\mathrm{h}_{0,j}\right)\right]$$

$$
\begin{aligned}
&\leq \frac{2dT\log 2}{\lambda} + \frac{1}{2\lambda}\log\sum_{j=1}^{n}\prod_{i=1}^{m}\Bigg[\frac{1}{2}\exp\Bigg(2\lambda\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\mathrm{x}_{\tau,i,j}\Bigg) \\
&\quad + \frac{1}{2}\exp\Bigg(-2\lambda\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)\mathrm{x}_{\tau,i,j}\Bigg)\Bigg] \\
&\quad + \frac{1}{2\lambda}\log\sum_{j=1}^{h}\prod_{i=1}^{m}\Bigg[\frac{1}{2}\exp\Bigg(2\lambda\Lambda_0^T\mathrm{h}_{0,j}\Bigg) + \frac{1}{2}\exp\Bigg(-2\lambda\Lambda_0^T\mathrm{h}_{0,j}\Bigg)\Bigg] \\
&\leq \frac{2dT\log 2}{\lambda} + \frac{1}{2\lambda}\log\sum_{j=1}^{n}\Bigg[\exp\Bigg(2\lambda^2\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)^2\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)^2\sum_{i=1}^{m}x_{\tau,i,j}^2\Bigg)\Bigg] \\
&\quad + \frac{1}{2\lambda}\log\sum_{j=1}^{h}\Bigg[\exp\Bigg(2\lambda^2\Lambda_0^{2T}\sum_{i=1}^{m}h_{0,j}^2\Bigg)\Bigg] \qquad (48c) \\
&\leq \frac{2dT\log 2}{\lambda} + \frac{\log n}{2\lambda} + \lambda\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)^2\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)^2 mB_{\mathbf{x}}^2 + \frac{\log h}{2\lambda} + \lambda\Lambda_0^{2T}m\|\mathbf{h}_0\|_\infty^2 \\
&\leq \frac{2dT\log 2 + \log\sqrt{n} + \log\sqrt{h}}{\lambda} + \lambda\Bigg(\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)^2\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)^2 mB_{\mathbf{x}}^2 + \Lambda_0^{2T}m\|\mathbf{h}_0\|_\infty^2\Bigg), \qquad (48d)
\end{aligned}
$$

where (48a) follows inequality (60), and (48b) holds by replacing with $\sum_{j=1}^{n}$ and $\sum_{j=1}^{h}$, respectively. In addition, (48c) follows (59), and (48d) is obtained by the following definition: At time step t, we define $\mathbf{X}_t \in \mathbb{R}^{n\times m}$, a matrix composed of m columns from the m input vectors $\{\mathbf{x}_{t,i}\}_{i=1}^{m}$; we also define $\|\mathbf{X}_t\|_{2,\infty} = \sqrt{\max_{k\in\{1,\ldots,n\}}\sum_{i=1}^{m}\mathrm{x}_{t,i,k}^2} \leq \sqrt{m}B_{\mathbf{x}}$, representing the maximum of the ℓ_2-norms of the rows of matrix $\mathbf{X}_t$, and $\|\mathbf{h}_0\|_\infty = \max_j |\mathrm{h}_{0,j}|$.

Choosing $\lambda = \sqrt{\frac{2dT\log 2+\log\sqrt{n}+\log\sqrt{h}}{\left(\sum_{l=1}^{d}\beta_l\Lambda_l\right)^2\left(\frac{\Lambda_0^T-1}{\Lambda_0-1}\right)^2 mB_{\mathbf{x}}^2+\Lambda_0^{2T}m\|\mathbf{h}_0\|_\infty^2}}$, we achieve the upper bound:

$$
\mathfrak{R}_S(\mathcal{F}_{d,T}) \leq \sqrt{\frac{2(4dT\log 2 + \log n + \log h)}{m}\Bigg(\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)^2\Big(\frac{\Lambda_0^T-1}{\Lambda_0-1}\Big)^2 B_{\mathbf{x}}^2 + \Lambda_0^{2T}\|\mathbf{h}_0\|_\infty^2\Bigg)}. \qquad (49)
$$

It can be noted that $\mathfrak{R}_S(\mathcal{F}_{d,T})$ in (49) is derived for the real-valued functions $\mathcal{F}_{d,T}$. For the vector-valued functions $\mathcal{F}_{d,T} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}^h$ (in Theorem 3.1), we apply the contraction lemma (Lemma D.1) to a Lipschitz loss to obtain the complexity of such vector-valued functions by means of the complexity of the real-valued functions. Specifically, in Theorem 3.1, under the assumption of the 1-Lipschitz loss function and from Theorem C.1, Lemma D.1, we complete the proof.

□

D.1 Comparison with existing generalization bounds

Recent works have established generalization bounds for RNN models with a single recurrent layer ($d = 1$) using Rademacher complexity [see FastRNN in Kusupati et al. (2018)] or PAC-Bayes theory [see SpectralRNN in Zhang et al. (2018)]. We re-state these generalization bounds below and apply Theorem 3.1 with $d = 1$ to compare with our bound for reweighted-RNN.

FastRNN (Kusupati et al., 2018). The hidden state $\mathbf{h}_t$ of FastRNN is updated as follows:

$$
\begin{aligned}
\tilde{\mathbf{h}}_t &= \phi(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t) \\
\mathbf{h}_t &= a\tilde{\mathbf{h}}_t + b\mathbf{h}_{t-1}, \qquad (50)
\end{aligned}
$$

where $0 \leq a, b \leq 1$ are trainable parameters parameterized by the sigmoid function. Under the assumption that $a + b = 1$, the Rademacher complexity $\mathfrak{R}_S(\mathcal{F}_T)$ of the class $\mathcal{F}_T$ of FastRNN (Kusupati et al., 2018), with $\|\mathbf{W}\|_F \leq \alpha_F$, $\|\mathbf{U}\|_F \leq \beta_F$, and $\|\mathbf{x}_t\|_2 \leq B$, is given by

$$\mathfrak{R}_S(\mathcal{F}_T) \leq \frac{2a}{\sqrt{m}} B\beta_F \Big(\frac{(1 + a(2\alpha_F - 1))^{T+1} - 1}{a(2\alpha_F - 1)} \Big). \tag{51}$$

Alternatively, under the additional assumption that $a \leq \frac{1}{2(2\alpha_F - 1)T}$, the bound in Kusupati et al. (2018) becomes:

$$\mathfrak{R}_S(\mathcal{F}_T) \leq \frac{2a}{\sqrt{m}} B\beta_F \Big(\frac{2a(2\alpha_F - 1)(T + 1) - 1}{(2\alpha_F - 1)} \Big). \tag{52}$$

SpectralRNN (Zhang et al., 2018). The hidden state $\mathbf{h}_t$ and output $\mathbf{y}_t \in \mathbb{R}^{n_\mathbf{y}}$ of SpectralRNN are computed as:

$$\begin{aligned} \mathbf{h}_t =& \phi(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t) \\ \mathbf{y}_t =& \mathbf{Y}\mathbf{h}_t, \end{aligned} \tag{53}$$

where $\mathbf{Y} \in \mathbb{R}^{n_\mathbf{y} \times h}$. The generalization error in Zhang et al. (2018) is derived for a classification problem. For any $\delta > 0, \gamma > 0$, with probability $\geq 1 - \delta$ over a training set S of size m, the generalization error (Zhang et al., 2018) of SpectralRNN is bounded by

$$\mathcal{O}\left(\sqrt{\frac{\frac{B^2 T^4 \xi \ln(\xi)}{\gamma^2} (\|\mathbf{W}\|_F^2 + \|\mathbf{U}\|_F^2 + \|\mathbf{Y}\|_F^2) \cdot \zeta + \ln \frac{m}{\delta}}{m}} \right), \tag{54}$$

where $\zeta = \max\{\|\mathbf{W}\|_2^{2T-2}, 1\} \max\{\|\mathbf{U}\|_2^2, 1\} \max\{\|\mathbf{Y}\|_2^2, 1\}$ and $\xi = \max\{n, n_\mathbf{y}, h\}$.

Reweighted-RNN. Based on Theorem 3.1, under the assumption that the initial hidden state $\mathbf{h}_0 = \mathbf{0}$, the Rademacher complexity of reweighted-RNN with $d = 1$ is bounded as

$$\mathfrak{R}_S(\mathcal{F}_{1,T}) \leq \sqrt{\frac{4T \log 2 + \log n + \log h}{m}} \Big(\sqrt{2}\beta_1 \frac{\alpha_1^T - 1}{\alpha_1 - 1} B_\mathbf{x} \Big). \tag{55}$$

We observe that the bound of SpectralRNN in (54) depends on T^2, whereas the bound of FastRNN either grows exponentially with T (51) or is proportional to T (52). Our bound (55) depends on $\sqrt{T}$, given that the second factor in (55) is only dependent on the norm constraints $\alpha_1,\ \beta_1$ and the input training data; meaning that it is tighter than those of SpectralRNN and FastRNN in terms of the number of time steps.

D.2 Background on Rademacher complexity calculus

The contraction lemma in Shalev-Shwartz & Ben-David (2014) gives the Rademacher complexity of the composition of a class of functions with ρ-Lipschitz functions.

Lemma D.1. *(Shalev-Shwartz & Ben-David, 2014, Lemma 26.9—Contraction lemma)*
Let $\mathcal{F}$ be a set of functions, $\mathcal{F} = \{f : \mathcal{X} \mapsto \mathbb{R}\}$, and $\Phi_1, ..., \Phi_m$, ρ-Lipschitz functions, namely, $|\Phi_i(\alpha) - \Phi_i(\beta)| \leq \rho|\alpha - \beta|$ for all $\alpha, \beta \in \mathbb{R}$ for some $\rho > 0$. For any sample set S of m points $\mathbf{x}_1, ..., \mathbf{x}_m \in \mathcal{X}$, let $(\Phi \circ f)(\mathbf{x}_i) = \Phi(f(\mathbf{x}_i))$. Then,

$$\frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\epsilon} \in \{\pm 1\}^m} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \epsilon_i (\Phi \circ f)(\mathbf{x}_i) \right] \leq \frac{\rho}{m} \mathop{\mathbb{E}}_{\boldsymbol{\epsilon} \in \{\pm 1\}^m} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \epsilon_i f(\mathbf{x}_i) \right], \tag{56}$$

alternatively, $\mathfrak{R}_S(\boldsymbol{\Phi} \circ \mathcal{F}) \leq \rho \mathfrak{R}_S(\mathcal{F})$, where $\boldsymbol{\Phi}$ denotes $\Phi_1(\mathbf{x}_1), ..., \Phi_m(\mathbf{x}_m)$ for S.

Proposition D.2. *(Mohri et al., 2018, Proposition A.1—Hölder's inequality)*
Let $p, q \geq 1$ be conjugate: $\frac{1}{p} + \frac{1}{q} = 1$. Then, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$\|\mathbf{x} \cdot \mathbf{y}\|_1 \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q, \tag{57}$$

with the equality when $|\mathrm{y}_i| = |\mathrm{x}_i|^{p-1}$ for all $i \in [1, n]$.

Supporting inequalities:

(i) If A, B are sets of positive real numbers, then:

$$\sup(\mathrm{AB}) = \sup(\mathrm{A}) \cdot \sup(\mathrm{B}). \tag{58}$$

(ii) Given $x \in \mathbb{R}$, we have:

$$\frac{\exp(x) + \exp(-x)}{2} \leq \exp(x^2/2). \tag{59}$$

(iii) Let X and Y be random variables, the Cauchy–Bunyakovsky–Schwarz inequality gives:

$$(\mathbb{E}[\mathrm{XY}])^2 \leq \mathbb{E}[\mathrm{X}^2] \cdot \mathbb{E}[\mathrm{Y}^2]. \tag{60}$$

(iv) If ψ is a convex function, the Jensen's inequality gives:

$$\psi(\mathbb{E}[\mathrm{X}]) \leq \mathbb{E}[\psi(\mathrm{X})]. \tag{61}$$