Generalization Error Bounds for Deep Unfolding RNNs

Boris Joukovsky [1,2] **Tanmoy Mukherjee** [1,2] **Huynh Van Luong** [1,2] **Nikos Deligiannis** [1,2]

[1]Department of Electronics and Informatics, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
[2]imec, Kapeldreef 75, B-3001 Leuven, Belgium

Abstract

Recurrent Neural Networks (RNNs) are powerful models with the ability to model sequential data. However, they are often viewed as black-boxes and lack in interpretability. Deep unfolding methods take a step towards interpretability by designing deep neural networks as learned variations of iterative optimization algorithms to solve various signal processing tasks. In this paper, we explore theoretical aspects of deep unfolding RNNs in terms of their generalization ability. Specifically, we derive generalization error bounds for a class of deep unfolding RNNs via Rademacher complexity analysis. To our knowledge, these are the first generalization bounds proposed for deep unfolding RNNs. We show theoretically that our bounds are tighter than similar ones for other recent RNNs, in terms of the number of timesteps. By training models in a classification setting, we demonstrate that deep unfolding RNNs can outperform traditional RNNs in standard sequence classification tasks. These experiments allow us to relate the empirical generalization error to the theoretical bounds. In particular, we show that over-parametrized deep unfolding models like reweighted-RNN achieve tight theoretical error bounds with minimal decrease in accuracy, when trained with explicit regularization.

1 INTRODUCTION

The past few years has seen an undeniable success of Recurrent Neural Networks (RNNs) in applications ranging from machine translation [Bahdanau et al., 2015] and image captioning [Karpathy and Fei-Fei, 2017] to speech processing [Oord et al., 2016]. Despite their impressive performance, existing approaches are typically designed based on heuristics of the task and rely on engineering experience. As a result, such models lack in theoretical understanding and interpretability. A step towards post-hoc interpretability of RNNs was made by Karpathy et al. [2015], where they observed the advantages of internal states of the long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] over traditional RNNs to model interpretable patterns in the input data; this study was further complemented by Goldberg [2015]. Subsequent works in explainable deep learning have also proposed methods to explain the decision-making process of a trained LSTM for a given prediction, in terms of the most influential inputs [Li et al., 2016, Arras et al., 2019].

Efforts toward a theoretical understanding of RNNs also involve the derivation of bounds on the generalization error, i.e., the difference between the empirical loss and the expected loss. Such bounds rely on measures—such as the Rademacher complexity [Bartlett and Mendelson, 2002], the PAC-Bayes theory [McAllester, 2003], the algorithm stability [Bousquet and Elisseeff, 2002] and robustness [Xu and Mannor, 2012]—and aim at understanding how accurately a model is able to generalise to unseen data in relation to the optimization algorithm used for training, the network architecture, or the underlying data structure.

Generalization error bounds (GEBs) for RNN models were studied by Arora et al. [2018], Wei and Ma [2019] and Akpinar et al. [2019]. The theoretical result by Arora et al. [2018] for low-dimensional embeddings using LSTMs models was derived for classification tasks. That work showed an interesting property of embeddings such as GloVe and word2vec, that is, forming a good sensing matrix for text leads to better representations than those obtained with random matrices. The data-dependent sample complexity of deep neural networks was studied by Wei and Ma [2019]. Unlike the existing Rademacher complexity bounds that depend on norms of the weight matrices and depend exponentially on the model depth [Bartlett and Mendelson, 2002, Dziugaite and Roy, 2017, Neyshabur et al., 2018], the study by Wei and Ma [2019] derived tighter Radermacher bounds with the consideration of additional data-dependant

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

properties controlling the norms of hidden layers and the norms of the Jacobians of each layer with respect to all previous layers. The derivation methods by Wei and Ma [2019] were extended to provide GEBs for RNNs scaling polynomial with respect to the depth of the model. Specifically, generalization abilities were explored by Akpinar et al. [2019], where explicit sample complexity bounds for single and multi-layer RNNs were derived.

The focus of this work is in studying how model interpretability—by means of designing networks based on deep unfolding—interacts with model generalization capacity. Deep unfolding methods design deep models as unrolled iterations of optimization algorithms; a key result in this direction is the learned iterative shrinkage thresholding algorithm (LISTA) [Gregor and LeCun, 2010] that solves the sparse coding problem. Iterative optimization algorithms are usually highly interpretable because they are developed via modeling the physical processes underlying the problem and/or capturing prior domain knowledge [Lucas et al., 2018]. Hence, deep networks designed by deep unfolding capture domain knowledge and promote model-based structure in the data; in other words, such networks can be naturally interpreted as a parameter optimized algorithm [Monga et al., 2019]. Deep unfolding has been used to develop interpretable RNNs: SISTA-RNN [Wisdom et al., 2017] is an extension of LISTA that solves a sequential signal reconstruction task based on sparse modelling with the aid of side-information. Alternatively, Le et al. [2019] designed ℓ_1-ℓ_1-RNN by unfolding a proximal method that solves an ℓ_1-ℓ_1 minimization problem [Mota et al., 2017]. In our recent work [Luong et al., 2021], we proposed reweighted-RNN, which unfolds a reweighted version of the ℓ_1-ℓ_1 minimization problem and incoporates additional weights so as to increase the model expressivity. Deep unfolding RNNs excel in solving the underlying signal reconstruction tasks, outperforming traditional RNN baselines while having a substantially lower parameter count than the LSTM or gated recurrent unit (GRU) [Cho et al., 2014] models; however, their ability to leverage the sparse structure of data as a means to solve traditional RNN tasks (e.g., sequence classification) is relatively unexplored. Furthermore, despite the progress in deep unfolding models, their generalization ability has received no such attention; particularly, no work exists studying the GEBs of deep unfolding RNN models. In this paper, we study theoretical aspects of deep unfolding RNNs by means of the GEBs. We also benchmark deep unfolding RNNs in classification problems and relate the empirical generalization error to the theoretical one.

The contributions of this work are as follows:

- We derive generalization error bounds (GEB) for deep unfolding RNNs by means of Rademacher complexity analysis, by taking the reweighted-RNN model [Luong et al., 2021] as a prototype, which also subsumes the ℓ_1-ℓ_1-RNN [Le et al., 2019]. We also present an extension of the GEBs when the RNN are used in a classification setting. We show theoretically that the proposed bounds are tighter than other recent lightweight RNNs with generalization guarantees, namely, SpectralRNN [Zhang et al., 2018] and FastGRNN [Kusupati et al., 2018], in terms of number of time steps T.
- We assess the performance of deep unfolding RNNs in classification settings, namely in speech command detection and language modelling tasks. For the first task, our results show that reweighted-RNN outperforms other lightweight RNNs, and that deep unfolding RNNs are competitive with traditional RNN baselines (i.e., vanilla RNN, LSTM and GRU) while being smaller in size. For the second task, we observe a significant improvement of reweighted-RNN over all other models.
- By taking speech command detection as an example, we show that the proposed GEB for reweighted-RNN is tight and agrees with the empirical generalization gap. This can be achieved when the model is sufficiently regularized, while maintaining high classification accuracy.

The remainder of this paper is as follows: Section 2 reviews traditional stacked RNN models, followed by an introduction to deep unfolding RNNs. Section 3 presents the proposed GEBs for deep unfolding RNNs, which is obtained by studying the complexity of their latent representation stage. The bound is then extended to the classification problem. In Section 4, we experimentally compare reweighted-RNN to other deep unfolding and traditional RNN models on classification tasks. We also evaluate the GEB experimentally and relate the theoretical aspects to empirical observations.

2 BACKGROUND TO RNN MODELS

2.1 STACKED RNNs

For each time step $t = 1, \ldots, T$, the vanilla RNN recursively updates the latent representations $\mathbf{h}_t$ through linear transformation of the input sample $\mathbf{x}_t$ and the previous state $\mathbf{h}_{t-1}$, followed by a non linear activation $\sigma(\cdot)$. To achieve higher expressivity, one may use a stacked vanilla RNN [Pascanu et al., 2014] by juxtaposing multiple RNN layers. In this setting, the representation $\mathbf{h}_t^{(l)}$ at layer l and time step t is computed by

$$\mathbf{h}_t^{(l)} = \begin{cases} \sigma\Big(\mathbf{V}_1 \mathbf{h}_{t-1}^{(l)} + \mathbf{U}_1 \mathbf{x}_t\Big), \text{ if } l = 1, \\ \sigma\Big(\mathbf{W}_l \mathbf{h}_t^{(l-1)} + \mathbf{V}_l \mathbf{h}_{t-1}^{(l)}\Big), \text{ if } l > 1, \end{cases} \tag{1}$$

where $\mathbf{U}_1, \mathbf{W}_l, \mathbf{V}_l$ are the weight matrices learned per layer l and $\sigma(\cdot)$ is the activation function. For sequence classification tasks, RNN models can be trained end-to-end with a

softmax-activated linear layer to obtain output class probabilities. It is recognized that the vanilla RNN is prone to gradient vanishing, resulting in training instabilities in the case of long data sequences and deep architectures. Popular RNN variants including LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014] alleviate this issue by leveraging gating mechanisms to better capture long-term dependencies in the data. More recent lightweight models include FastGRNN by Kusupati et al. [2018] and SpectralRNN [Zhang et al., 2018] make use of weight factorization to achieve low parameter count; such models improve the stability of deeper RNNs and are highly competitive with standard RNNs on various prediction tasks.

2.2 DEEP UNFOLDING RNNs

Unlike traditional RNNs, the considered deep unfolding RNNs follow a signal reconstruction model with low-complexity data priors. How these models generalize to classification tasks, e.g. by training a linear classifier on the latent representation $\mathbf{h}_T^{(d)}$, remains an open problem. In what follows, we review our recent reweighted-RNN [Luong et al., 2021] that unfolds a proximal algorithm to solve a generic sequential signal reconstruction problem.

2.2.1 Reweighted-RNN

Problem Formulation: Consider the problem of reconstructing a sequence of signals $\mathbf{s}_t \in \mathbb{R}^{n_0}$, $t = 1, 2, \ldots, T$, from a sequence of noisy measurement vectors $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \boldsymbol{\eta}_t$ where $\mathbf{x}_t \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n \times n_0}$ ($n \ll n_0$). According to domain knowledge in sparse signal reconstruction with side information and video-frame reconstruction [Deligiannis et al., 2013, Mota et al., 2017], it is assumed that: (i) $\mathbf{s}_t$ 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, $\mathbf{s}_t = \mathbf{D}\mathbf{h}_t$; and (ii) the difference of two successive sparse representations is also sparse up to an affine transform $\mathbf{P} \in \mathbb{R}^{h \times h}$ promoting the temporal correlation, i.e, $\mathbf{h}_t - \mathbf{P}\mathbf{h}_{t-1}$ is small in the ℓ_1-norm sense. Assumptions (i) and (ii) define a sequential ℓ_1-ℓ_1 reconstruction problem; it is however recognized that ℓ_1-ℓ_1 minimization can be improved by reweighted minimization [Candès et al., 2008, Van Luong et al., 2018], which leads to sparser representations. The reweighting scheme introduces the positive weights $\mathbf{g} \in \mathbb{R}^h$ to individually penalize the magnitude of each element of $\mathbf{h}_t$ and $\mathbf{h}_t - \mathbf{P}\mathbf{h}_{t-1}$. To further generalize, a reweighing matrix $\mathbf{Z} \in \mathbb{R}^{h \times h}$ is introduced as an extra transformation of $\mathbf{h}_t$. The considered reweighted ℓ_1-ℓ_1 minimization problem is:

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

Algorithm 1: Reweighted ℓ_1-ℓ_1 algorithm for sequential signal reconstruction.

Input: Measurements $\mathbf{x}_1, \ldots, \mathbf{x}_T$, measurement matrix $\mathbf{A}$, dictionary $\mathbf{D}$, affine transform $\mathbf{P}$, 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.
Output: Sequence of sparse codes $\mathbf{h}_1, \ldots, \mathbf{h}_T$.
for *t = 1,…,T* **do**
  $\mathbf{h}_t^{(0)} = \mathbf{P}\mathbf{h}_{t-1}^{(d)}$
  for *l = 1 to d* **do**
    $\mathbf{u} = [\mathbf{Z}_l - \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}]\mathbf{h}_t^{(l-1)} + \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{x}_t$
    $\mathbf{h}_t^{(l)} = \Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \mathbf{P}\mathbf{h}_{t-1}^{(d)}}(\mathbf{u})$
  end
end
return $\mathbf{h}_1^{(d)}, \ldots, \mathbf{h}_T^{(d)}$

The first term in Problem (2) corresponds to the data fidelity term and $\lambda_1, \lambda_2 > 0$ are regularization parameters controlling the weighted-ℓ_1 penalization terms. Note that the presence of $\mathbf{Z}$ in Problem (2) is a deviation from the reweighted minimization scheme of Candès et al. [2008]. Nevertheless, the study in [Luong et al., 2021] shows that $\mathbf{Z}$ enables the unfolded RNN to achieve higher expressivity by following more closely the vanilla RNN architecture.

Proximal Algorithm: We use a proximal gradient method [Beck and Teboulle, 2009] to solve Problem (2), which is given in Algorithm 1. At iteration l, the algorithm performs a proximal gradient update in two steps: first, by updating $\mathbf{h}_t^{(l-1)}$ using the gradient of the fidelity term of Problem (2). Second, it applies the proximal operator $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \boldsymbol{\hbar}}(\mathbf{u})$ element-wise to the result, with $\boldsymbol{\hbar} \equiv \mathbf{P}\mathbf{h}_{t-1}$ denoting the side-information from the previous time-step.

The closed-form solution of the proximal operator is given in the supplementary material and we refer to Luong et al. [2021] for extended proofs. Fig. 1 depicts the proximal operators for $\hbar \geq 0$, which is essentially a 2-plateaus soft-thresholding function. Notice that each entry of $\mathbf{g}_l$ controls the width of the two thresholds, independently for each element. The plateau at height $\hbar$ in Fig. 1 promotes the correlation of the signal with the side information.

Deep Unfolded RNN: The reweighted-RNN model [Luong et al., 2021] is build by unrolling Algorithm 1 across the iterations $l = 1, \ldots, d$ (yielding the hidden layers) and time steps $t = 1, \ldots, T$. The proximal operator $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l, \frac{\lambda_2}{c}\mathbf{g}_l, \boldsymbol{\hbar}}(\mathbf{u})$ also serves as the non-linear activation function. The form of this function depends on the learnable parameters λ_1, λ_2, c, and $\mathbf{g}_l$, $\forall l = 1, \ldots, d$, allowing reweighted-RNN to learn

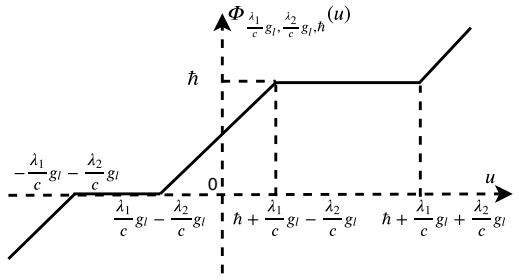

Figure 1: The generic form of the proximal operator $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l,\frac{\lambda_2}{c}\mathbf{g}_l,\hbar}(u)$ for Algorithm 1 (particular case of $\hbar \geq 0$) - but also the activation function in the reweighted-RNN. Note that per unit per layer $\mathbf{g}_l$ leads to a different activation function.

per-neuron adjustable activations. In summary, the layers of reweighted-RNN realise the following updates:

$$\mathbf{h}_t^{(l)} = \begin{cases} \Phi_{\frac{\lambda_1}{c}\mathbf{g}_1,\frac{\lambda_2}{c}\mathbf{g}_1,\mathbf{P}\mathbf{h}_{t-1}^{(d)}}\Big(\mathbf{W}_1\mathbf{h}_{t-1}^{(d)}+\mathbf{U}_1\mathbf{x}_t\Big), & \text{if } l=1, \\ \Phi_{\frac{\lambda_1}{c}\mathbf{g}_l,\frac{\lambda_2}{c}\mathbf{g}_l,\mathbf{P}\mathbf{h}_{t-1}^{(d)}}\Big(\mathbf{W}_l\mathbf{h}_t^{(l-1)}+\mathbf{U}_l\mathbf{x}_t\Big), & \text{if } l>1, \end{cases} \tag{3}$$

where $\mathbf{U}_l$, $\mathbf{W}_1$, $\mathbf{W}_l$ are defined as

$$\mathbf{U}_l = \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}, \forall l, \tag{4}$$

$$\mathbf{W}_1 = \mathbf{Z}_1\mathbf{P} - \frac{1}{c}\mathbf{Z}_1\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D}\mathbf{P}, \tag{5}$$

$$\mathbf{W}_l = \mathbf{Z}_l - \frac{1}{c}\mathbf{Z}_l\mathbf{D}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{D},\ l>1. \tag{6}$$

and the trainable parameters of the model are $\mathbf{\Theta} = \{\mathbf{A}, \mathbf{D}, \mathbf{P}, \mathbf{h}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_d, \mathbf{g}_1, \ldots, \mathbf{g}_d, c, \lambda_1, \lambda_2\}$ with the parameters $\mathbf{A}, \mathbf{D}$ and $\mathbf{P}$ corresponding to the sensing matrix, the sparsifying dictionary and the correlation matrix, respectively. An illustration of reweighted-RNN is given in Fig. 2(a), in contrast to the stacked vanilla RNN in Fig. 2(b): it can be observed that reweighted-RNN is characterized by skip-connections since the input $\mathbf{x}_t$ is incorporated into each layer, while the side information $\mathbf{P}\mathbf{h}_{t-1}^{(l)}$ modulates the activation function $\Phi_{\frac{\lambda_1}{c}\mathbf{g}_l,\frac{\lambda_2}{c}\mathbf{g}_l,\boldsymbol{\hbar}}$.

2.2.2 ℓ_1-ℓ_1-RNN

ℓ_1-ℓ_1-RNN was proposed by Le et al. [2019] as a deep unfolding RNN that learns the iterations of a proximal algorithm to solve an ℓ_1-ℓ_1 minimization problem, that is, Problem (2) with $\mathbf{Z} = \mathbf{I}$ and $\mathbf{g} = \mathbf{1}$. Thus, ℓ_1-ℓ_1-RNN can be seen as instance of reweighted-RNN with update equations obtained by setting $\mathbf{Z}_1, \ldots, \mathbf{Z}_d = \mathbf{I}$ and $\mathbf{g}_1, \ldots, \mathbf{g}_d = \mathbf{1}$ in (3), (4), (5), (6). As a result, one layer of ℓ_1-ℓ_1-RNN

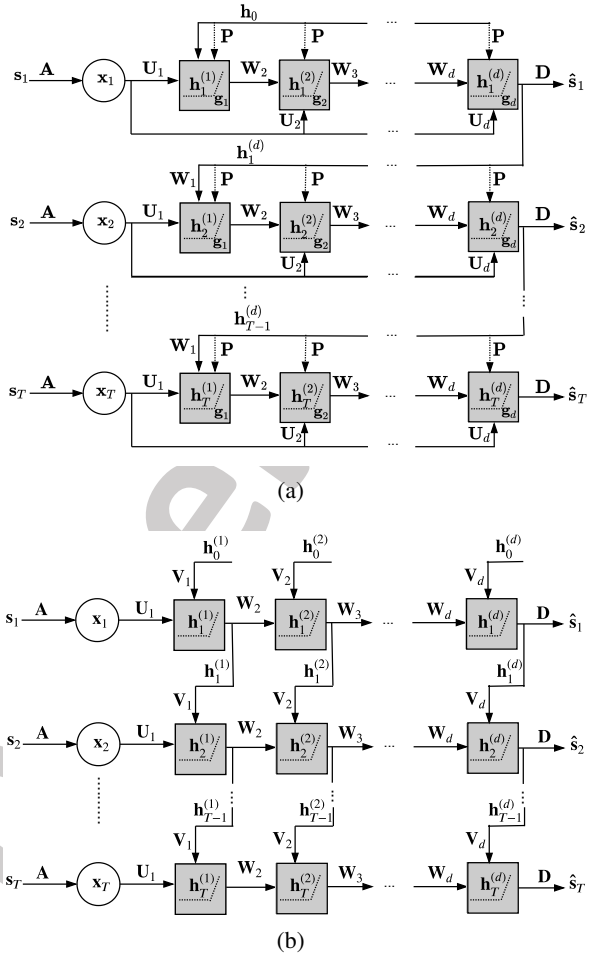

Figure 2: The architecture of (a) reweighted-RNN vis-à-vis (b) the vanilla stacked RNN [Pascanu et al., 2014].

updates $\mathbf{h}_t^{(l)}$ according to:

$$\mathbf{h}_t^{(l)} = \begin{cases} \phi_{\frac{\lambda_1}{c},\frac{\lambda_2}{c},\mathbf{P}\mathbf{h}_{t-1}^{(d)}}\Big(\mathbf{W}_1\mathbf{h}_{t-1}^{(d)}+\mathbf{U}_1\mathbf{x}_t\Big), & \text{if } l=1, \\ \phi_{\frac{\lambda_1}{c},\frac{\lambda_2}{c},\mathbf{P}\mathbf{h}_{t-1}^{(d)}}\Big(\mathbf{W}_2\mathbf{h}_t^{(l-1)}+\mathbf{U}_1\mathbf{x}_t\Big), & \text{if } l>1, \end{cases} \tag{7}$$

with

$$\mathbf{U}_1 = \frac{1}{c}\mathbf{D}^T\mathbf{A}^T, \tag{8}$$

$$\mathbf{W}_1 = \mathbf{P} - \frac{1}{c}\mathbf{D}^T\mathbf{A}^T\mathbf{A}\mathbf{D}\mathbf{P}, \tag{9}$$

$$\mathbf{W}_2 = \mathbf{I} - \frac{1}{c}\mathbf{D}^T\mathbf{A}^T\mathbf{A}\mathbf{D}. \tag{10}$$

ℓ_1-ℓ_1-RNN and reweighted-RNN have similar update equations and activation functions, to the difference that the absence of a trainable weight $\mathbf{Z}_l$ in ℓ_1-ℓ_1-RNN leads to shared weights $\mathbf{W}_l$ and $\mathbf{U}_l$, starting from $l = 2$, reducing the overall expressivity of the model. Also, the shape of its activation function cannot be controlled per element, since $\mathbf{g}_l$ is fixed to the all-ones.

3 GENERALIZATION ERROR BOUNDS FOR THE DEEP UNFOLDING RNNs

We briefly present the theoretical background regarding Rademacher complexity and the related generalization error bound (GEB). We then propose an upper bound of the empirical Rademacher complexity of reweighted-RNN, along with a similar bound for ℓ_1-ℓ_1-RNN which is subsumed by the former. This bound is derived for the latent representation stage. We then formulate a GEB for the same models when used in a classification setting, i.e., by appending a single-layer linear classification layer to the models.

Notation: Let $f_{\boldsymbol{\mathcal{W}}}^{(d)} : \mathbb{R}^n \mapsto \mathbb{R}^h$ be the function computed by a d-layer network with weight parameters $\boldsymbol{\mathcal{W}}$. The network $f_{\boldsymbol{\mathcal{W}}}^{(d)}$ 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., $\mathbf{y}_i = f_{\boldsymbol{\mathcal{W}}}^{(d)}(\mathbf{x}_i)$. 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:

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim\mathcal{D}}\big[\ell\big(f(\mathbf{x}_i), \mathbf{y}_i\big)\big], \tag{11}$$

and

$$L_S(f) = \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim S}\big[\ell\big(f(\mathbf{x}_i), \mathbf{y}_i\big)\big]. \tag{12}$$

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

Definition 3.1 (Empirical Rademacher Complexity)**.** *Let $\mathcal{F}$ be a hypothesis set of functions (e.g., neural networks). The empirical Rademacher complexity of $\mathcal{F}$ [Shalev-Shwartz and Ben-David, 2014] for a training sample set S is defined as follows:*

$$\mathfrak{R}_S(\mathcal{F}) = \frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\in\{\pm1\}^m}\left[\sup_{f\in\mathcal{F}}\sum_{i=1}^{m}\epsilon_i f(\mathbf{x}_i)\right], \tag{13}$$

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

The GEB for the functional family $\mathcal{F}$ is derived based on the Rademacher complexity in the following theorem:

Theorem 3.2. *[Shalev-Shwartz and 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 $1 - \delta$,*

$$L_{\mathcal{D}}(f) - L_S(f) \leq 2\mathfrak{R}_S(\ell \circ \mathcal{F}) + 4\eta\sqrt{\frac{2\log(4/\delta)}{m}}. \tag{14}$$

It can be remarked that the bound in (14) depends on the training set S, which is able to be applied to a number of learning problems, e.g., regression and classification, given a loss function ℓ.

3.1 DERIVED GENERALIZATION ERROR BOUNDS

The following theorem presents an upper bound on the empirical Rademacher complexity of the family of functions given by reweighted-RNN:

Theorem 3.3 (Empirical Rademacher Complexity of reweighted-RNN)**.** *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 $\|\mathbf{X}_t\|_{2,\infty} \leq \sqrt{m}B_{\mathbf{x}}$, and an initial hidden state $\mathbf{h}_0$. Then,*

$$\mathfrak{R}_S(\mathcal{F}_{d,T}) \leq \sqrt{\frac{2(4dT\log 2 + \log n + \log h)}{m}} \cdot \sqrt{\Big(\sum_{l=1}^{d}\beta_l\Lambda_l\Big)^2\Big(\frac{\Lambda_0^T - 1}{\Lambda_0 - 1}\Big)^2 B_{\mathbf{x}}^2 + \Lambda_0^{2T}\|\mathbf{h}_0\|_\infty^2}, \tag{15}$$

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

The proof of Theorem 3.3 is given in the supplementary material. The Rademacher complexity in Theorem 3.3 can be used in combination with the GEB of Theorem 3.2 for any choice of Lipschitz-continuous and bounded loss function. If the complexity measure is small, the network can be learned with a small generalization error. The bound in (15) is in the order of the square root of the network depth d multiplied by the number of time steps T. It also 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 (15) 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.

We also derive a similar bound for ℓ_1-ℓ_1-RNN considering that it is an instance of reweighted-RNN (cf. Section 2.2.2)

Corollary 3.3.1 (The empirical Rademacher complexity of ℓ_1-ℓ_1-RNN)**.** *Let $\mathcal{F}_{d,T} : \mathbb{R}^h \times \mathbb{R}^n \mapsto \mathbb{R}^h$ denote the functional class of ℓ_1-ℓ_1-RNN with T time steps, where $\|\mathbf{W}_1\|_{1,\infty} \leq \alpha_1$, $\|\mathbf{W}_2\|_{1,\infty} \leq \alpha_2$, $\|\mathbf{U}_1\|_{1,\infty} \leq \beta_1$, and $1 \leq l \leq d$. Assume that the input data $\|\mathbf{X}_t\|_{2,\infty} \leq \sqrt{m}B_{\mathbf{x}}$,*

initial hidden state $\mathbf{h}_0$*. Then,*

$$\mathfrak{R}_S(\mathcal{F}_{d,T}) \leq \sqrt{\frac{2(4dT\log 2 + \log n + \log h)}{m}} \cdot \sqrt{\beta_1^2\Big(\frac{\alpha_2^d - 1}{\alpha_2 - 1}\Big)^2\Big(\frac{\alpha_1^T\alpha_2^{T(d-1)} - 1}{\alpha_1\alpha_2^{(d-1)} - 1}\Big)^2 B_{\mathbf{x}}^2 + \alpha_1^{2T}\alpha_2^{2T(d-1)}\|\mathbf{h}_0\|_\infty^2}. \quad (16)$$

By contrasting (15) with (16) under the assumption of constrained weights, we see that the complexity of ℓ_1-ℓ_1-RNN has a polynomial dependence on α_1, β_1 and α_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.

A comparison can be made between our bounds and the ones proposed by Zhang et al. [2018] for SpectralRNN and Kusupati et al. [2018] for FastRNN (the non-gated version of FastGRNN). Overall, all bounds, including ours, exhibit a T-exponential dependency on the weight norms, which suggests that regularization mechanisms must be used to to achieve low model capacity and tight generalization guarantees. If we assume that weights are bounded so as to neglect the T-exponential dependency, the GEB of SpectralRNN is still dependent on T^2. Likewise, Kusupati et al. [2018] shows that the GEB of FastRNN can be made linear in T by restricting the range of some of the model's parameters. With constrained weights, our GEBs depends on $\sqrt{T}$, meaning that our bounds are tighter in terms of the number of time steps.

3.2 EXTENSION TO CLASSIFICATION

We present an extension of our bounds for the linear classification task with c classes using the final latent vector $\mathbf{h}_T^{(d)}$ as input. We particularize the GEB for the ramp loss r_γ, which is paramaterized by $\gamma > 0$ and defined as

$$r_\gamma(x) = \begin{cases} 0, & x < -\gamma, \\ 1 + \frac{x}{\gamma}, & -\gamma \leq x \leq 0, \\ 1, & 0 < x. \end{cases} \quad (17)$$

In practice, we evaluate r_γ on a data sample $\mathbf{y}$ with ground-truth label c using the classification margin $\mathcal{M}(\mathbf{y}, c) = \mathbf{y}[c] - \max_{c' \neq c}\mathbf{y}[c']$ by computing $r_\gamma(-\mathcal{M}(\mathbf{y}, c))$. For convenience, we define $\ell_\gamma^c(\mathbf{y}) \equiv r_\gamma(-\mathcal{M}(\mathbf{y}, c))$. Intuitively, $\ell_\gamma^c(\mathbf{y})$ penalizes samples that are classified incorrectly or correctly with low margin. It can be noted that $\ell_\gamma^c(\mathbf{y})$ is uniformly bounded by $\eta = 1$ and is $\frac{1}{\gamma}$-Lipschitz, which is required by the assumptions of Theorem 3.2.

We also assume that the classifier output $\mathbf{y}$ is given by $\mathbf{Y}\mathbf{h}_T^{(d)} \equiv y(\mathbf{h}_t^{(d)})$ with $\mathbf{Y} \in \mathbb{R}^{c \times h}$; it follows that $y(\mathbf{h}_T^{(d)})$ is ρ-Lipschitz on its input with $\rho = \min(\max_i \|\mathbf{Y}_i\|_2, \max_i \|\mathbf{Y}_i\|_1)$ where $\mathbf{Y}_i$ denotes the i^{th} row of $\mathbf{Y}$. Following these properties and by using the contraction lemma [Shalev-Shwartz and Ben-David, 2014], Theorem 3.3 can be reformulated into the following proposition:

Proposition 3.4 (Generalization Error Bound Extended to the RNN with Linear Classifier)**.** *Consider the functional family of classifiers* $\mathcal{F} = \{\ell_\gamma^c \circ y \circ f : \mathcal{X} \mapsto \mathbb{R} \mid f \in \mathcal{F}_{d,T}\}$ *obtained by composing reweighted-RNN (or* ℓ_1*-*ℓ_1*-RNN) with the linear classifier* $\mathbf{y} = \mathbf{Y}\mathbf{h}_T^{(d)}$ *and with the* ℓ_γ^c *loss. Consider* $\rho = \min(\max_i \|\mathbf{Y}_i\|_2, \max_i \|\mathbf{Y}_i\|_1)$ *where* $\mathbf{Y}_i$ *denotes the* i^{th} *row of* $\mathbf{Y}$*. Then,*

$$L_{\mathcal{D},\gamma}(f) - L_{S,\gamma}(f) \leq \frac{2\rho}{\gamma}\mathfrak{R}_S(\mathcal{F}_{d,T}) + 4\sqrt{\frac{2\log(4/\delta)}{m}}. \quad (18)$$

4 EXPERIMENTAL RESULTS

In this section, we benchmark deep unfolding RNNs, specifically reweighted-RNN, ℓ_1-ℓ_1-RNN and SISTA-RNN, on a speech command classification task using the Google-12 and Google-30 datasets [Warden, 2018] and on a language modelling task using the Wikitext2 dataset [Merity et al., 2016]. A comparison is made with various RNN models including the vanilla RNN, LSTM and GRU baselines, and the lightweight SpectralRNN and FastGRNN models. Although deep unfolding RNNs have been traditionally used for signal reconstruction tasks, our experiments show promising results in solving traditional RNN tasks, with reweighted-RNN achieving superior performance. Finally, we relate the experimental generalization gap observed on the Google-30 dataset with the derived GEB using the ℓ_γ^c loss defined in Section 3.2 and an appropriate weight regularization scheme. Our theoretical bound is shown to follow the empirical generalization performance of reweighted-RNN.

4.1 EXPERIMENTAL SETUP

Concerning reweighted-RNN, the overcomplete dictionary $\mathbf{D}$ is initialized according to a random uniform distribution, the weights $\mathbf{Z}_l, ..., \mathbf{Z}_d$ and $\mathbf{P}$ are initialized to the identity $\mathbf{I}$ and the reweighing vectors $\mathbf{g}_1, ..., \mathbf{g}_d$ to the all-ones vectors. We initialize $\{c, \lambda_1, \lambda_2\}$ to $\{1.0, 0.003, 0.02\}$. The measurement operator $\mathbf{A}$ is fixed to $\mathbf{I}$. We add a trainable weight matrix $\mathbf{Y} \in \mathbb{R}^{n_c \times h}$ to obtain classification scores based on the last hidden state according to $\mathbf{y} = \mathbf{Y}\mathbf{h}_T^{(d)}$ with n_c corresponding to the number of classes (or the number of tokens in the case of language modeling). For all models in general, we use $h = 100$ hidden units and varying hidden layers $d = \{1, 2, 3, 4, 5, 6\}$. Models are trained in PyTorch by optimizing the cross-entropy loss using the Adam optimizer. The initial learning rate is set to 3×10^{-4} and follows a step decay policy with a factor 0.3 every 50 epochs, except

Table 1: Test accuracy on the Google-12 dataset for different network depths d. (a): RNN baselines, (b): lightweight RNNs, (c): deep unfolding RNNs

	Model	d=1	d=2	d=3	d=4	d=5	d=6
(a)	RNN	88.60	89.40	88.78	89.46	90.48	90.86
	LSTM	90.42	91.30	92.24	91.30	92.02	91.46
	GRU	91.07	**92.05**	**92.28**	**92.24**	92.15	90.94
(b)	SpectralRNN	89.80	91.20	91.20	91.20	91.50	91.80
	FastGRNN	**91.21**	91.34	91.42	91.53	91.66	91.72
(c)	SISTA-RNN	90.14	90.88	91.12	91.50	92.02	92.12
	ℓ_1-ℓ_1-RNN	90.27	91.02	91.34	91.78	92.14	92.20
	reweighted-RNN	90.49	91.30	91.23	91.66	**92.37**	**92.41**

Table 2: Test accuracy on the Google-30 dataset for different network depths d. (a): RNN baselines, (b): lightweight RNNs, (c): deep unfolding RNNs

	Model	d=1	d=2	d=3	d=4	d=5	d=6
(a)	RNN	35.26	38.27	45.28	47.80	48.90	54.27
	LSTM	92.14	**93.83**	93.69	93.80	92.68	92.48
	GRU	91.60	92.77	92.85	92.51	92.86	91.82
(b)	SpectralRNN	91.30	91.81	92.00	92.17	93.50	93.72
	FastGRNN	89.34	88.65	90.68	91.51	92.56	93.51
(c)	SISTA-RNN	92.18	92.50	93.17	93.22	93.64	93.78
	ℓ_1-ℓ_1-RNN	92.32	92.78	93.05	93.70	93.80	93.92
	reweighted-RNN	**92.51**	92.95	**93.97**	**93.81**	**94.10**	**94.00**

for GRU and LSTM which use an initial learning rate of 3×10^{-3}. The batch size is set to 100 and models are trained for 200 epochs. While deep unfolding RNNs have their own layer-stacking schemes derived from unfolding minimization algorithms, we use the stacking rule in [Pascanu et al., 2014] to build deep networks for other RNN architectures.

4.2 SPEECH COMMAND DETECTION

The Google-12 and Google-30 datasets [Warden, 2018] contain utterances of short speech commands with background noise belonging to 12 and 30 classes, respectively. We adopt the same featurization technique of Kusupati et al. [2018], consisting of the standard log Mel-filter-bank featurization with 32 filters on $T = 99$ time steps.

We report our test classification accuracies in Tables 1 and 2. For Google-12, reweighted-RNN outperforms both the traditional RNN baselines and the lightweight models with 5 and 6 layers, whereas GRU achieves superior accuracy for lower number of layers. For Google-30, reweighted-RNN achieves higher performance almost systematically. The improved accuracy of reweighted-RNN is due to the additional learnable weights $\mathbf{Z}_l$ and the per-neuron adaptive activations, which increase the expressivity of the model. The results are also in line with the ablation study in [Luong et al., 2021], which demonstrated that reweighted-RNN improves video-frame reconstruction over other deep unfolding RNNs and traditional ones. Although standard RNNs outperform deep unfolding RNNs in some configurations, deep unfolding architectures are still interesting in terms of model size, especially compared to the well-performing GRU and LSTM

Table 3: Test perplexity on the Wikitext dataset for different network depths d. (a): RNN baselines, (b): lightweight RNNs, (c): deep unfolding RNNs

	Model	d=1	d=2	d=3	d=4	d=5	d=6
(a)	RNN	145.6	140.3	137.3	134.2	122.3	121.2
	LSTM	127.3	119.8	118.3	121.4	112.3	110.4
	GRU	143.4	112.2	117.0	108.0	112.6	109.4
(b)	SpectralRNN	69.1	68.5	67.4	65.8	63.1	59.8
	FastGRNN	69.3	69.2	67.8	66.2	66.1	64.7
(c)	SISTA-RNN	128.2	121.6	116.3	109.3	94.3	88.7
	ℓ_1-ℓ_1-RNN	67.7	67.3	**65.6**	63.7	59.5	56.6
	reweighted-RNN	**67.1**	**65.4**	67.5	**59.5**	**56.4**	**52.1**

models. For $d = 3$ layers, we observe that reweighted-RNN, ℓ_1-ℓ_1-RNN and SISTA-RNN have 48K, 17K and 8K parameters respectively, while FastGRNN and SpectralRNN have 43K and 50K parameters respectively. Finally, the vanilla RNN, LSTM and GRU models have respectively 61K, 223K and 169K parameters.

4.3 LANGUAGE MODELLING TASK

Language modeling (LMs) serves as a foundation for natural language processing. The task involves predicting the $n+1$ token given a history of n such tokens. Trained LMs have been applied to speech recognition [Yu and Deng, 2014], machine translation [Cho et al., 2014], natural language generation and as a feature extractor for general downstream tasks [Peters et al., 2018]. LMs act at multiple levels of granularities, i.e., at word, sub-word or character level. While the objective remains the same, each of them adds a layer of complexity and challenges. We test the different RNNs on the Wikitext2 dataset [Merity et al., 2016], a commonly used dataset consisting of 2 million tokens. As the Wikipedia articles are relatively long, capturing long range dependencies is key to achieve reasonable performance. Table 3 shows the test perplexity obtained for number of hidden layers d. Similar to the previous experiments, we observe that reweighted-RNN outperforms the other considered RNN models.

4.4 ERROR BOUND EVALUATION

We perform an empirical evaluation of the generalization error bound (GEB) for the classification task on the Google-30 dataset. It can be observed that the GEB based on the Rademacher complexity of reweighted-RNN in (15) depends on the T-exponential of the norms of $\mathbf{W}_l$. However, in Section 4.2, models are trained without explicit regularization of the weight's norm, leading to potentially high GEBs that do not provide useful guarantees on the performance of the model on unseen data. Consequently, we introduce an explicit regularization term in the training loss function by multiplying the maximum ℓ_1-norm of the rows of $\mathbf{W}_l$ with a regularization parameter λ to ensure that the T-exponential

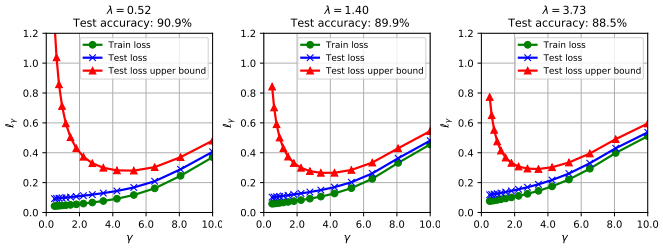

Figure 3: Evaluation of the error bound on the ramp loss for various values of γ, for three regularization levels λ, for reweighted-RNN trained on the Google-30 dataset.

dependency on the weights becomes negligible. In practice, we train a single layer reweighted-RNN model using three different regularization levels $\lambda = \{0.52, 1.40, 3.73\}$ and using the categorical cross-entropy as the base loss, according to the experimental protocol described in Section 4.1. The classification GEB (18) is then calculated for the ℓ_γ^c loss for different margins γ and for $\delta = 0.05$ (i.e., a probability of 95% that the bound is met).

Empirical evaluations of the GEB are reported in Fig. 3 along with the respective losses observed in practice on the training and test sets. For small values of γ, the theoretical GEB is close to the unity (note that ℓ_γ^c cannot exceed 1 in practice). By increasing γ, the empirical generalization gap decreases and the theoretical bound becomes tighter. Around $\gamma = 4.0$, the theoretical bound on the test error is minimal. For very large values of γ, the empirical training and testing errors become much larger as a result of the large margin γ; even if the bound is tighter, this setting is less indicative of the true performance of the model. Nevertheless, for reasonable choices of γ, our GEB is tight and proves that our model has good generalization properties with small test loss.

As shown in Table 4, the accuracy of reweighted-RNN slightly decreases with higher regularization levels, with a decrease of 2.4% in test accuracy from $\lambda = 0.52$ to $\lambda = 3.73$; it is however necessary to perform weight regularization to ensure that the theoretical GEB is tight and informative. Moreover, it was found that ℓ_1-ℓ_1-RNN is hardly able to maintain a sufficiently high test accuracy while keeping the theoretical GEB in an acceptable range. In Table 4, the test accuracy of ℓ_1-ℓ_1-RNN drops to 73.5% for the upper bound on $\mathfrak{R}_S$ to lead to a sufficiently tight GEB. This further confirms that reweighted-RNN offers more flexibility than ℓ_1-ℓ_1-RNN as a result of the additional learnable weights $\mathbf{Z}_l$ and per-neuron learnable activations, while ensuring good generalization abilities according to the GEB (18).

5 CONCLUSIONS

In this paper, we have explored theoretical aspects of deep unfolding RNNs by establishing generalization error bounds

Table 4: Evaluation of the Rademacher complexity and test accuracy of the single-layer reweighted-RNN and ℓ_1-ℓ_1-RNN models trained on the Google-30 dataset for various regularization levels

Model		Test acc. (%)	$\mathfrak{R}_S(\mathcal{F}_{d,T})$ (upper bound)
reweighted-RNN	$\lambda = 0.52$	90.9	2.92×10^{-1}
	$\lambda = 1.40$	89.9	1.83×10^{-1}
	$\lambda = 3.73$	88.5	1.61×10^{-1}
ℓ_1-ℓ_1**-RNN**	$\lambda = 0.52$	84.8	5.45×10^{14}
	$\lambda = 1.40$	82.4	3.59×10^{6}
	$\lambda = 3.73$	73.5	3.94×10^{0}

obtained via Rademacher complexity analysis of a class of deep unfolding RNNs, namely, the reweighted-RNN [Luong et al., 2021] and ℓ_1-ℓ_1-RNN [Le et al., 2019] models. For sufficiently constrained weights, the GEB is polynomially dependent on the norm of the weights and the overparametrization scheme of reweighted-RNN, allowing for a finer control of the bound. Moreover, our proposed bounds are tighter than similar ones obtained for state-of-the-art lightweight RNN models, in terms of time steps T. Furthermore, our experiments have demonstrated that deep unfolding RNNs can outperform state-of-the-art traditional RNNs in classification problems, leading to interesting perspectives for deep unfolding RNNs outside of their original application fields. These experiments also show that it is possible to tighten the gap between the theoretical bound of reweighted-RNN and the empirical generalization error, if the model is sufficiently regularized while not affecting its accuracy.

6 ACKNOWLEDGEMENT

We would like to thank the reviewers for their comments which helped improve the manuscript. This research received funding from FWO, Belgium (research project G093817N and PhD fellowship strategic basic research 1SB5721N), and Innoviris, Belgium (REFINED Project).

References

Nil-Jana Akpinar, Bernhard Kratzwald, and Stefan Feuerriegel. Sample complexity bounds for recurrent neural networks with application to combinatorial graph problems. *arXiv preprint arXiv:1901.10289*, 2019.

Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and LSTMs. In *International Conference on Learning Representations*, 2018.

Leila Arras, José Arjona-Medina, Michael Widrich, Grégoire Montavon, Michael Gillhofer, Klaus-Robert Müller,

Sepp Hochreiter, and Wojciech Samek. Explaining and interpreting LSTMs. In *Explainable AI: Interpreting, explaining and visualizing deep learning*, pages 211–238. Springer, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*, 2015.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2: 499–526, 2002.

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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Nikos Deligiannis, Adrian Munteanu, Shuang Wang, Samuel Cheng, and Peter Schelkens. Maximum likelihood Laplacian correlation channel estimation in layered wyner-ziv coding. *IEEE Transactions on Signal Processing*, 62(4):892–904, 2013.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

Yoav Goldberg. The unreasonable effectiveness of character-level language models, 2015. URL `https://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139`.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

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

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39 (4):664–676, April 2017.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

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.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

Alice Lucas, Michael Iliadis, Rafael Molina, and Aggelos K. Katsaggelos. Using deep neural networks for inverse problems in imaging: Beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

Huynh Van Luong, Boris Joukovsky, and Nikos Deligiannis. Designing interpretable recurrent neural networks for video reconstruction via deep unfolding. *IEEE Transactions on Image Processing*, 30:4099–4113, 2021.

David McAllester. Simplified PAC-Bayesian margin bounds. In *Learning theory and Kernel machines*, pages 203–215. Springer, 2003.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *arXiv preprint arXiv:1912.10557*, 2019.

João FC Mota, Nikos Deligiannis, and Miguel RD Rodrigues. Compressed sensing with prior information: Strategies, geometry, and bounds. *IEEE Transactions on Information Theory*, 63(7):4472–4496, 2017.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. *International Conference on Learning Representations*, 2018.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. In *International Conference on Learning Representations*, 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, June 2018.

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

Huynh Van Luong, Nikos Deligiannis, Jürgen Seiler, Søren Forchhammer, and Andre Kaup. Compressive online robust principal component analysis via n-ℓ_1 minimization. *IEEE Transactions on Image Processing*, 27(9): 4314–4329, 2018.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via Lipschitz augmentation. In *Advances in Neural Information Processing Systems 32*, pages 9725–9736, 2019.

Scott Wisdom, Thomas Powers, James Pitton, and Les Atlas. Building recurrent networks by unfolding iterative thresholding for sequential sparse recovery. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4346–4350. IEEE, 2017.

Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014. ISBN 1447157788.

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*, 2018.