Learning to Adaptively Scale Recurrent Neural Networks

Hao Hu,[1] **Liqiang Wang,**[1] **Guo-Jun Qi**[2]
[1]University of Central Florida
[2]Huawei Cloud
{haohu, lwang}@cs.ucf.edu, Guojun.qi@huawei.com

Abstract

Recent advancements in recurrent neural network (RNN) research have demonstrated the superiority of utilizing multiscale structures in learning temporal representations of time series. Currently, most of multiscale RNNs use fixed scales, which do not comply with the nature of dynamical temporal patterns among sequences. In this paper, we propose Adaptively Scaled Recurrent Neural Networks (ASRNN), a simple but efficient way to handle this problem. Instead of using predefined scales, ASRNNs are able to learn and adjust scales based on different temporal contexts, making them more flexible in modeling multiscale patterns. Compared with other multiscale RNNs, ASRNNs are bestowed upon dynamical scaling capabilities with much simpler structures, and are easy to be integrated with various RNN cells. The experiments on multiple sequence modeling tasks indicate ASRNNs can efficiently adapt scales based on different sequence contexts and yield better performances than baselines without dynamical scaling abilities.

Introduction

Recurrent Neural Networks (RNNs) play a critical role in sequential modeling as they have achieved impressive performances in various tasks (Campos et al. 2017)(Chang et al. 2017)(Chung, Ahn, and Bengio 2016)(Neil, Pfeiffer, and Liu 2016). Yet learning long-term dependencies from long sequences still remains a very difficult task (Bengio, Simard, and Frasconi 1994) (Hochreiter et al. 2001)(Ye et al. 2017)(Hu et al. 2017). Among various ways that try to handle this problem, modeling multiscale patterns seem to be a promising strategy since many multiscale RNN structures perform better than other non-scale modeling RNNs in multiple applications (Koutnik et al. 2014)(Neil, Pfeiffer, and Liu 2016)(Chung, Ahn, and Bengio 2016)(Chang et al. 2017)(Campos et al. 2017)(Chang et al. 2014). Multiscale RNNs can be roughly divided into two groups based on their design philosophies. The first group trends to modeling scale patterns with the hierarchical architectures and prefixed scales for different layers. This may lead to at least two disadvantages. First, the prefixed scale can not be adjusted to fit the temporal dynamics throughout the time. Although patterns in different scale levels require distinct frequencies to update, they do not always stick to a certain scale and could vary at different time steps. For example, in polyphonic music modeling, distinguishing different music styles demands RNNs to model various emotion changes throughout music pieces. While emotion changes are usually controlled by the lasting time of notes, it is insufficient to model such patterns using only fixed scales as the notes last differently at different time. Secondly, stacking multiple RNN layers greatly increases the complexity of the entire model, which makes RNNs even harder to train. Unlike this, another group of multiscale RNNs models scale patterns through gate structures (Neil, Pfeiffer, and Liu 2016)(Campos et al. 2017)(Qi 2016). In such cases, additional control gates are learned to optionally update hidden for each time step, resulting in a more flexible sequential representations. Yet such modeling strategy may not remember information which is more important for future outputs but less related to current states.

In this paper, we aim to model the underlying multiscale temporal patterns for time sequences while avoiding all the weaknesses mentioned above. To do so, we present Adaptively Scaled Recurrent Neural Networks (ASRNNs), a simple extension for existing RNN structures, which allows them to adaptively adjust the scale based on temporal contexts at different time steps. Using the causal convolution proposed by (Van Den Oord et al. 2016), ASRNNs model scale patterns by firstly convolving input sequences with wavelet kernels, resulting in scale-related inputs that parameterized by the scale coefficients from kernels. After that, scale coefficients are sampled from categorical distributions determined by different temporal contexts. This is achieved by sampling Gumbel-Softmax (GM) distributions instead, which are able to approximate true categorical distributions through the re-parameterization trick. Due to the differentiable nature of GM, ASRNNs could learn to flexibly determine which scale is most important to target outputs according to temporal contents at each time step. Compared with other multiscale architectures, the proposed ASRNNs have several advantages. First, there is no fixed scale in the model. The subroutine for scale sampling can be trained to select proper scales to dynamically model the temporal scale patterns. Second, ASRNNs can model multiscale patterns within a single RNN layer, resulting in a much simpler structure and easier optimization process. Besides, ASRNNs do not use gates to control the updates of hidden states. Thus there is no risk of missing information for future outputs.

To verify the effectiveness of ASRNNs, we conduct extensive experiments on various sequence modeling tasks, including low density signal identification, long-term memorization, pixel-to-pixel image classification, music genre recognition and language modeling. Our results suggest that ASRNNs can achieve better performances than their non-adaptively scaled counterparts and are able to adjust scales according to various temporal contents. We organize the rest paper like this: the first following section reviews relative literatures, then we introduce ASRNNs with details in next section; after that the results for all evaluations are presented, and the last section concludes the paper.

Related Work

As a long-lasting research topic, the difficulties of training RNNs to learn long-term dependencies are considered to be caused by several reasons. First, the gradient exploding and vanishing problems during back propagation make training RNNs very tough (Bengio, Simard, and Frasconi 1994) (Hochreiter et al. 2001). Secondly, RNN memory cells usually need to keep both long-term dependencies and short-term memories simultaneously, which means there should always be trade-offs between two types of information. To overcome such problems, some efforts aim to design more sophisticated memory cell structures. For example, Long-short term memory (LSTM) (Hochreiter and Schmidhuber 1997) and gated recurrent unit (GRU) (Chung et al. 2014), are able to capture more temporal information; while some others attempt to develop better training algorithms and initialization strategies such as gradient clipping (Pascanu, Mikolov, and Bengio 2013), orthogonal and unitary weight optimization (Arjovsky, Shah, and Bengio 2016)(Le, Jaitly, and Hinton 2015) (Wisdom et al. 2016)(Qi, Hua, and Zhang 2009)(Wang et al. 2016)(Qi, Aggarwal, and Huang 2012) etc. These techniques can alleviate the problem to some extent (Tang et al. 2017)(Li et al. 2017)(Wang et al. 2012).

Meanwhile, previous works like (Koutnik et al. 2014) (Neil, Pfeiffer, and Liu 2016) (Chung, Ahn, and Bengio 2016) (Tang et al. 2007) (Hua and Qi 2008) suggest learning temporal scale structures is also the key to this problem. This stands upon the fact that temporal data usually contains rich underlying multiscale patterns (Schmidhuber 1991)(Mozer 1992) (El Hihi and Bengio 1996) (Lin et al. 1996) (Hu and Qi 2017). To model multiscale patterns, a popular strategy is to build hierarchical architectures. These RNNs such as hierarchical RNNs (El Hihi and Bengio 1996), clockwork RNNs (Koutnik et al. 2014) and Dilated RNNs (Chang et al. 2017) etc, contain hierarchical architectures whose neurons in high-level layers are less frequently updated than those in low-level layers. Such properties fit the natures of many latent multiscale temporal patterns where low-level patterns are sensitive to local changes while high-level patterns are more coherent with the temporal consistencies. Instead of considering hierarchical architectures, some multiscale RNNs model scale patterns using control gates to decide whether to update hidden states or not at a certain time step. Such structures like phased LSTMs (Neil, Pfeiffer, and Liu 2016) and skip RNNs (Campos et al. 2017), are able to adjust their modeling scales based on current temporal contexts, leading to more reasonable and flexible sequential representations. Recently, some multiscale RNNs like hierarchical multi-scale RNNs (Chung, Ahn, and Bengio 2016), manage to combine the gate-controlling updating mechanism into hierarchical architectures and has made impressive progress in language modeling tasks. Yet they still employ multi-layer structures which make the optimization not be easy.

Adaptively Scaled Recurrent Neural Networks

In this section we introduce Adaptively Scaled Recurrent Neural Networks (ASRNNs), a simple but useful extension for various RNN cells that allows to dynamically adjust scales at each time step. An ASRNNs is consist of three components: scale parameterization, adaptive scale learning and RNN cell integration, which will be covered in following subsections.

Scale Parameterization

We begin our introduction for ASRNNs with scale parameterization. Suppose $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 \cdots, \mathbf{x}_T]$ is an input sequence where $\mathbf{x}_t \in \mathcal{R}^n$. At time t, instead of taking only the current frame $\mathbf{x}_t$ as input, ASRNNs compute an alternative scale-related input $\tilde{\mathbf{x}}_t$, which can be obtained by taking a causal convolution between the original input sequence $\mathbf{X}$ and a scaled wavelet kernel function ϕ_{j_t}.

More specifically, let J be the number of considered scales. Consider a wavelet kernel ϕ of size K. At any time t, given a scale $j_t \in \{0, \cdots, J-1\}$, the input sequence $\mathbf{X}$ is convolved with a scaled wavelet kernel $\phi_{j_t} = \phi(\frac{i}{2^{j_t}})$. This yields the following scaled-related input $\tilde{\mathbf{x}}_t$ at time t

$$\tilde{\mathbf{x}}_t = (\mathbf{X} * \phi_{j_t})_t = \sum_{i=0}^{2^{j_t}K-1} \mathbf{x}_{t-i}\phi(\frac{i}{2^{j_t}}) \in \mathcal{R}^n \qquad (1)$$

where for any $i \in \{t - 2^{j_t}K + 1, \cdots, t-1\}$, we manually set $\mathbf{x}_i = \mathbf{0}$ iff $i \leq 0$. And the causal convolution operator $*$ (Van Den Oord et al. 2016) is defined to avoid the resultant $\tilde{\mathbf{x}}_t$ depending on future inputs. We also let $\phi(\frac{i}{2^{j_t}}) = 0$ iff $2^{j_t} \nmid i$. It is easy to see that $\tilde{\mathbf{x}}_t$ can only contain information from $\mathbf{x}_{t-i}$ when $i = 2^{j_t}k, k \in \{1, \cdots, K\}$. In other words, there are skip connections between $\mathbf{x}_{t-2^{j_t}(k-1)}$ and $\mathbf{x}_{t-2^{j_t}k}$ in the scale j_t. While j_t becomes larger, the connections skip further.

It is worth mentioning that the progress for obtaining scale-related input $\tilde{\mathbf{x}}_t$ is quite similar as the convolutions with the real waveforms in (Van Den Oord et al. 2016). By stacking several causal convolutional layers, (Van Den Oord et al. 2016) is able to model temporal patterns in multiple scale levels with its exponential-growing receptive field. However, such abilities are achieved through a hierarchical structure where each layer is given a fixed dilation factor that does not change through out time. To avoid this, we replace the usual convolution kernels with wavelet kernels, which come with scaling coefficients just like j_t in equation 1. By varying j_t, $\tilde{\mathbf{x}}_t$ is allowed to contain information from different scale levels. Thus we call it scale parameterization. We further demonstrate it's possible to adaptively control j_t based on temporal contexts through learning, which will be discussed in subsection .

Adaptive Scale Learning

To adjust scale j_t at different time t, we need to sample j_t from a categorical distribution where each class probability is implicitly determined by temporal contexts. However, it is impossible to directly train such distributions along with deep neural networks because of the non-differentiable nature of their discrete variables. Fortunately, (Jang, Gu, and Poole 2016) (Maddison, Mnih, and Teh 2016) propose Gumbel-Softmax (GM) distribution, a differentiable approximation for a categorical distribution that allow gradients to be back propagated through its samples. Moreover, GM employs the re-parameterization trick, which divides the distribution into a basic independent random variable and a deterministic function. Thus, by learning the function only, we can bridge the categorical sampling with temporal contexts through a differentiable process.

Now we introduce the process of learning to sample scale j_t with more details. Suppose $\boldsymbol{\pi}_t = [\pi_0^t, \cdots, \pi_{J-1}^t] \in [0,1]^J$ are class probabilities for scale set $\{0, \cdots, J-1\}$ and $\mathbf{z}_t = [z_0^t, \cdots, z_{J-1}^t] \in \mathcal{R}^J$ are some logits related to temporal contexts at time t. The relationship between $\boldsymbol{\pi}_t$ and $\mathbf{z}_t$ can be written as

$$\pi_i^t = \frac{\exp(z_i^t)}{\sum_{i'=0}^{J-1} \exp(z_{i'}^t)} \tag{2}$$

where $i \in \{0, \cdots, J-1\}$. Let $\mathbf{y}_t = [y_0^t, \cdots, y_{J-1}^t] \in [0,1]^J$ be a sample from GM. Based on (Jang, Gu, and Poole 2016), y_i^t for $i = 0, \cdots, J-1$ can be calculated as

$$y_i^t = \frac{\exp((\log \pi_i^t + g_i)/\tau)}{\sum_{i'=0}^{J-1} \exp((\log \pi_{i'}^t + g_{i'})/\tau)} \tag{3}$$

where $g_0, \cdots, g_{J-1}$ are i.i.d. samples drawn from the basic Gumbel$(0,1)$ distribution and τ controls how much the GM is close to a true categorical distribution. In other words, as τ goes to 0, $\mathbf{y}_t$ would become $\mathbf{j}_t$, the one-hot vector whose j_tth value is 1.

Thus with GM, it is clear that the sampled $\mathbf{j}_t$ is approximated by a differentiable function of $\mathbf{z}_t$. We further define $\mathbf{z}_t$ with the hidden states $\mathbf{h}_{t-1} \in \mathcal{R}^m$ and input $\mathbf{x}_t \in \mathcal{R}^n$ as

$$\mathbf{z}_t = \mathbf{W}_z \mathbf{h}_{t-1} + \mathbf{U}_z \mathbf{x}_t + \mathbf{b}_z \in \mathcal{R}^J \tag{4}$$

where $\mathbf{W}_z, \mathbf{U}_z$ are weight matrices and $\mathbf{b}_z$ is bias vector. Combing equations 2, 3 and 4, we achieve our goal of dynamically changing j_t by sampling from GM distributions that parameterized by $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$. Since the entire procedure is differentiable, matrices $\mathbf{W}_z$ and $\mathbf{U}_z$ can be optimized along with other parameters of ASRNNs during the training.

Integrating with Different RNN Cells

With both the techniques introduced in previously introduced two subsections, we are ready to incorporate the proposed adaptive scaling mechanism with different RNN cells, resulting in various forms of ASRNNs. Since both $\tilde{\mathbf{x}}_t$ and sampling for j_t don't rely on any specific memory cell designs, it's straightforward to do so by replacing original input frames $\mathbf{x}_t$ with $\tilde{\mathbf{x}}_t$. For example, a ASRNN with LSTM cells can be represented as

$$\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t = \text{sigmoid}(\mathbf{W}_{f,i,o}\mathbf{h}_{t-1} + \mathbf{U}_{f,i,o}\tilde{\mathbf{x}}_t + \mathbf{b}_{f,i,o}) \in \mathcal{R}^m \tag{5}$$

$$\mathbf{g}_t = \tanh(\mathbf{W}_g \mathbf{h}_{t-1} + \mathbf{U}_g \tilde{\mathbf{x}}_t + \mathbf{b}_g) \in \mathcal{R}^m \tag{6}$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \tag{8}$$

while a ASRNN with GRU cells can be written as

$$\mathbf{z}_t, \mathbf{r}_t = \text{sigmoid}(\mathbf{W}_{z,r}\mathbf{h}_{t-1} + \mathbf{U}_{z,r}\tilde{\mathbf{x}}_t + \mathbf{b}_{z,r}) \in \mathcal{R}^m \tag{9}$$

$$\mathbf{g}_t = \tanh(\mathbf{W}_g(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{U}_g \tilde{\mathbf{x}}_t + \mathbf{b}_g) \in \mathcal{R}^m \tag{10}$$

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \circ \mathbf{g}_t \tag{11}$$

where $\mathbf{W}_*, \mathbf{U}_*$ are weight matrices and $\mathbf{b}_*$ are bias vectors, and $\circ$ means element-wise multiplication. For rest of this paper, we use ASLSTMs to refer those integrated with LSTM cells, ASGRUs for those integrated with GRU cells and so on and so forth. We still call them ASRNNs when there is no specified cell types. It is also worth mentioning that a conventional RNN cell is the special case of its ASRNN counterpart when $J = K = 1$.

Discussion

Finally, we briefly analyze the advantages of ASRNNs over other multiscale RNN structures. As mentioned in section , there are many RNNs, including hierarchical RNNs (El Hihi and Bengio 1996) and Dilated RNNs (Chang et al. 2017) etc, that apply hierarchical architectures to model multiscale patterns. Compared to them, the advantages of ASRNNs are clear. First, ASRNNs are able to model patterns with multiple scale levels within a single layer, making their spatial complexity much lower than hierarchical structures. Although hierarchical models may reduce the neuron numbers for each layer to have the similar size as single layer ASRNNs, they are harder to train with deeper structures. What's more, compared with the fixed scales for different layers, adapted scales are easier to capture underlying patterns as they can be adjusted based on temporal contexts at different time steps.

Besides, other multiscale RNN models like phased LSTMs (Neil, Pfeiffer, and Liu 2016) and skip RNNs (Campos et al. 2017) etc, build gate structures to manage scales. Such gates are learned to determine whether to remember the incoming information at each time. However, this may lose information which is important for future time but not for current time. This problem would never happen to ASRNNs as according to equation 1, the current input $\mathbf{x}_t$ will always be included in $\tilde{\mathbf{x}}$ and $\mathbf{h}_t$ is updated every step. Thus there is no risk for ASRNNs to lose critical information. This is an important property especially for tasks with frame

Table 1: Accuracies for ASRNNs and baselines .

ACCURACY (%)	RNN	SRNN	ASRNN
LSTM	81.3	83.6	97.7
GRU	84.1	88.1	98.0

labels. In such cases previously irrelevant information may become necessary for later frame outputs. Thus information from every frame should be leveraged to get correct outputs at different time.

Experiments

In this section, we evaluate the proposed ASRNNs with five sequence modeling tasks: low density signal type identification, copy memory problem, pixel-to-pixel image classification, music genre recognition and word level language modeling. We also explore how the scales would be adapted along time. Unless specified otherwise, all the models are implemented using Tensorflow library (Abadi et al. 2016). We train all the models with the RMSProp optimizer (Tieleman and Hinton) and set learning rate and decay rate to 0.001 and 0.9, respectively. It is worth mentioning that there is no techniques such as recurrent batch norm (Semeniuta, Severyn, and Barth 2016) and gradient clipping (Pascanu, Mikolov, and Bengio 2013) applied during the training. All the weight matrices are initialized with glorot uniform initialization (Glorot and Bengio 2010). For ASRNNs, we choose Haar wavelet as default wavelet kernels, and set τ of Gumbel-Softmax to 0.1. We integrate ASRNNs with two popular RNN cells, LSTM (Hochreiter and Schmidhuber 1997) and GRU (Chung et al. 2014) and use their conventional counterparts as common baselines. Besides, the baselines also include scaled RNNs (SRNNs), a simplified version that every j_t is set to $J-1$. Additional baselines for individual tasks will be stated in the corresponding subsections if there are. For both SRNNs and ASRNNs, The maximal considered scale J and wavelet kernel size K are set to 4 and 8, respectively.

Low Density Signal Type Identification

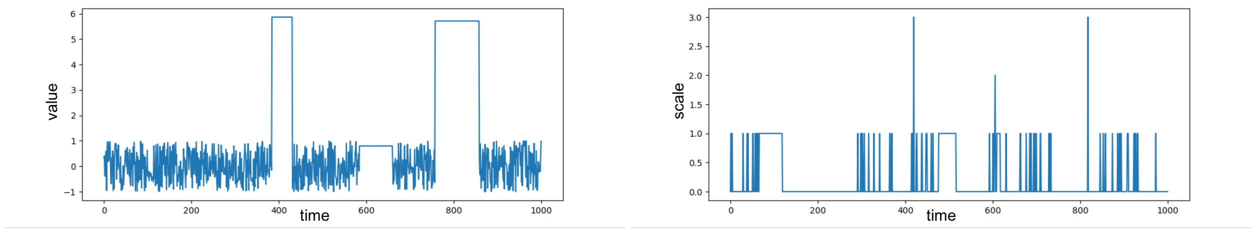
We begin our evaluation for ASRNNs with some synthetic data. The first task is low density signal type identification, which demands RNNs to distinguish the type of a long sequence that only contains limited useful information. More specifically, consider a sequence with length of 1000, first we randomly choose p subsequences at arbitrary locations of the sequence where $p \in \{3,4,5\}$. Each subsequence has different length T where $T \in \mathbb{Z}^+ \cap [20, 100]$ and we make sure that subsequences don't overlap with each other. For one sequence, all of its subsequences belong to one of the three types of waves: square wave, saw-tooth wave and sine wave, but with different amplitude A sampled from $[-7, 7]$. The rests of the sequence are filled with random noises sampled from $(-1, 1)$. The target is to identify which type of wave a sequence contains. Apparently, a sequence carries only $6\% \sim 50\%$ useful information, requiring RNNs capable of locating it efficiently.

Following above criterion, we randomly generate 2000 low density sequences for each type. We choose 1600 sequences per type for training and the remaining are for testing. Table 1 demonstrates the identification accuracies for baselines and ASRNNs. We can see the accuracies of both ASLSTM and ASGRU are over 97.5%, meaning they have correctly identified the types for most of sequences without being moderated by noise. Considering the much lower performance of baselines, it's confident to say that ASRNNs are able to efficiently locate useful information with adapted scales. Besides, we also observe there are similar patterns among some waves and their scale variation sequences. Figure 1 gives such an example, from which we see the scale 0 and 1 are more related to noises while the scale 2 and 3 only appear in the region with square form information. Moreover, the subsequence where the scale 2 is located is harder to identify as its values are too close to the noise. We believe such phenomena implies the scale variations could reflect some certain aspects that are helpful for understanding underlying temporal patterns.

Copy Memory Problem

Next we revisit the copy memory problem, one of the original LSTM tasks proposed by (Hochreiter and Schmidhuber 1997) to test the long-term dependency memorization abilities for RNNs. We closely follow the experimental setups used in (Arjovsky, Shah, and Bengio 2016) (Wisdom et al. 2016). For each input sequence with $T+20$ elements, The first ten are randomly sampled from integers 0 to 7. Then the rest of elements are all set to 8 except the $T+10$th to 9, indicating RNNs should begin to replicate the first 10 elements from now on. The last ten values of output sequence should be exactly the same as the first ten of the input. Cross entropy loss is applied for each time step. In addition to common baselines, we also adopt the memoryless baseline proposed by (Arjovsky, Shah, and Bengio 2016). The cross entropy of this baseline is $\frac{10\log(8)}{T+20}$, which means it always predict 8 for first $T+10$ steps while give a random guess of 0 to 7 for last 10 steps. For each T, we generate 10000 samples to train all RNN models.

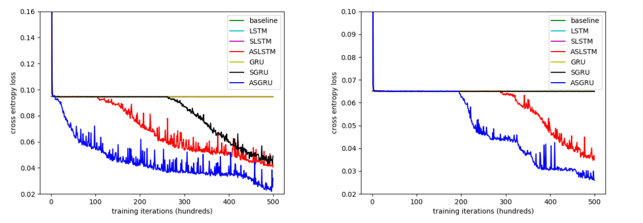
Figure 2 demonstrates the cross entropy curves of baselines and ASRNNs. We notice that both LSTM and GRU get stuck at the same cross entropy level with the memoryless baseline during the entire training process for both $T = 200$ and $T = 300$, indicating both LSTM and GRU are incapable of solving the problem with long time delays. This also agrees with the results reported in (Graves, Wayne, and Danihelka 2014) and (Arjovsky, Shah, and Bengio 2016). For SRNNs, it seems like fixed scales are little helpful since only the SGRU at $T = 200$ can have a lower entropy after 250 hundred steps. Unlike them, cross entropies of ASRNNs are observed to further decrease after certain steps of staying with baselines. Especially for $T = 200$, ASGRU almost immediately gets the entropy below the baseline with only a few hundreds of iterations passed. Besides, comparing figure 2a and 2b, ASGRUs are more resistant to the increasing of T as ASLSTMs need more time to wait before they can further reduce cross entropies. Overall, such behaviors prove



(a) A square wave sample.

(b) The corresponding scale variations.

Figure 1: The similar patterns between a raw square wave and its scale variations.



(a) $T = 200$

(b) $T = 300$

Figure 2: Cross entropies for copy memory problem. Best viewed in colors.

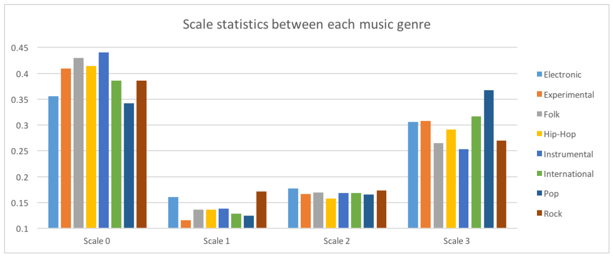


Figure 3: Statistics of scale selections between each music genre. The height of each bar indicates the ratio of how much times the scale is selected in the corresponding genre. Best viewed in colors.

ASRNNs have stronger abilities for memorizing long-term dependencies than baselines.

Pixel-to-Pixel Image Classification

Now we proceed our evaluation for ASRNNs with real world data. In this subsection, we study the pixel-to-pixel image classification problem using MNIST benchmark (LeCun et al. 1998). Initially proposed by (Le, Jaitly, and Hinton 2015), it reshapes all 28×28 images into pixel sequences with length of 784 before fed into RNN models, resulting in a challenge task where capturing long term dependencies is critical. We follow the standard data split settings and only feed outputs from the last hidden state to a linear classifier (Xing, Pei, and Keogh 2010). We conduct experiments for both unpermuted and permuted settings.

Table 2 summarizes results of all experiments for pixel-to-pixel MNIST classifications. The first two blocks are the comparisons between common baselines and ASRNNs with different cell structures. Their numbers of weights are adjusted to keep approximately same in order to be compared fairly. We also include other state-of-the-art results of single layer RNNs in the third block. It is easy to see that both SRNNs and ASRNNs achieve better performances than conventional RNNs with scale-related inputs on both settings. This is probably because causal convolutions between inputs and wavelet kernels can be treated as a spatial convolutional layer, allowing SRNNs and ASRNNs to leverage information that is spatially local but temporally remote. Moreover, the adapted scales help ASRNNs further reach the state-of-the-art performances by taking dilated convolutions with those pixels that more spatially related to the current position. It is also worth mentioning the proposed dynamical scaling is totally compatible with the techniques from the third part of the table 2 such as recurrent batch normalization (Cooijmans et al. 2016) and recurrent skip coefficients (Zhang et al. 2016). Thus ASRNNs can also benefit from them as well.

Music Genre Recognition

The next evaluation mission for ASRNNs is music genre recognition (MGR), a critical problem in the music information retrieval (MIR) (McKay and Fujinaga 2006) which requires RNNs to characterize the similarities between music tracks across many aspects such as cultures, artists and ages. Compared to other acoustic modeling tasks like speech recognition, MGR is considered to be more difficult as the boundaries between genres are hard to distinguish due to different subjective feelings among people (Scaringella, Zoia, and Mlynek 2006). We choose free music archive (FMA) dataset (Defferrard et al. 2017) to conduct our experiments. More specifically, we use the FMA-small, a balanced FMA subset containing 8000 music clips that distributed across 8 genres, where each clip lasts 30 seconds with sampling rate of 44100 Hz. We follow the standard $80/10/10\%$ data splitting protocols to get training, validation and test sets. We compute 13-dimensional log-mel frequency features (MFCC) with 25ms windows and 10ms frame steps for each clip, resulting in very long sequences with about 3000 entries. Besides, inspired by recent success of (Van Den Oord et al. 2016) and (Sainath et al. 2015), we are also encouraged to directly employ raw audio waves as inputs. Due to

Table 2: Classification accuracies for pixel-to-pixel MNIST. N stands for the number of hidden states. Italic numbers are results reported in the original papers. Bold numbers are best results for each part. ACC=accuracy, UNP/PER=unpermuted/permuted.

RNN	N	# OF WEIGHTS	MIN. SCALE	MAX. SCALE	AVG. SCALE	UNP ACC(%)	PER ACC(%)
LSTM	129	$\approx$ 68K	0	0	0	97.1	89.3
SLSTM	129	$\approx$ 68K	3	3	3	97.4	87.7
ASLSTM	128	$\approx$ 68K	0	3	0.92	**98.3**	**90.8**
GRU	129	$\approx$ 51K	0	0	0	96.4	90.1
SGRU	129	$\approx$ 51K	3	3	3	97.0	89.8
ASGRU	128	$\approx$ 51K	0	3	0.75	**98.1**	**91.2**
TANH-RNN (LE, JAITLY, AND HINTON 2015)	100	-	-	-	-	*35.0*	*33.0*
uRNN (ARJOVSKY, SHAH, AND BENGIO 2016)	512	$\approx$ 16K	-	-	-	*95.1*	*91.4*
FULL-CAPACITY uRNN (WISDOM ET AL. 2016)	512	$\approx$ 270K	-	-	-	*96.9*	*94.1*
iRNN (LE, JAITLY, AND HINTON 2015)	100	-	-	-	-	*97.0*	*82.0*
SKIP-LSTM (CAMPOS ET AL. 2017)	110	-	-	-	-	*97.3*	-
SKIP-GRU (CAMPOS ET AL. 2017)	110	-	-	-	-	*97.6*	-
sTANH-RNN (ZHANG ET AL. 2016)	64	-	-	-	-	*98.1*	*94.0*
RECURRENT BN-RNN (COOIJMANS ET AL. 2016)	100	-	-	-	-	**99.0**	**95.4**

Table 3: Music genre recognition on FMA-small. N stands for the number of hidden states. ACC=accuracy.

FEATURES	METHODS	N	# OF WEIGHTS	MIN. SCALE	MAX. SCALE	AVG. SCALE	ACC(%)
MFCC	LSTM	129	$\approx$ 74K	0	0	0	37.1
	SLSTM	129	$\approx$ 74K	3	3	3	37.7
	ASLSTM	128	$\approx$ 74K	0	3	1.34	**40.9**
	GRU	129	$\approx$ 56K	0	0	0	38.2
	SGRU	129	$\approx$ 56K	3	3	3	38.5
	ASGRU	128	$\approx$ 56K	0	3	1.39	**42.4**
	MFCC+GMM (AUCOUTURIER AND PACHET 2002)	-	-	-	-	-	21.3
RAW	LSTM	129	$\approx$ 68K	0	0	0	18.5
	SLSTM	129	$\approx$ 68K	3	3	3	18.9
	ASLSTM	128	$\approx$ 68K	0	3	1.47	**20.1**
	GRU	129	$\approx$ 51K	0	0	0	18.8
	SGRU	129	$\approx$ 51K	3	3	3	18.4
	ASGRU	128	$\approx$ 51K	0	3	1.59	**19.5**
	RAW+CNN (DIELEMAN AND SCHRAUWEN 2014)	-	-	-	-	-	17.5

limited computational resources, we have to reduce the sampling rate to 200 Hz for raw music clips while resultant sequences are still two times longer than MFCC sequences.

We demonstrate all the MGR results on FMA-small in the Table 3. Besides RNN models, we also include two baselines without temporal modeling abilities (GMM for MFCC and CNN for raw). We can see when using MFCC features, both the ASLSTM and ASGRU can outperform SRNNs and their conventional counterparts with about $3 \sim 4\%$ improvements. This is an encouraging evidence to show how adapted scales can boost the modeling capabilities of RNNs for MGR. However, the recognition accuracies drop significantly for all models when applying raw audio waves as inputs. In such cases, the gains from adapted scales are marginal for both the ASLSTM and ASGRU. We believe it is due to the low sampling rate for raw music clips since too much information is lost. However, increasing sampling rate will significantly rise the computational costs and make it eventually prohibitive for training RNNs.

To further understand the patterns behind such variations, we do statistics on how many times a scale has been selected for each genre, which is normalized and illustrated in figure 3. In general, all genres prefer to choose scale 0 and 3 since their ratio values are significantly higher than the other two. However, there are also obvious differences between genres within the same scale. For example, instrumental music tracks have more steps with scale 0 than Pop musics, while it's completely opposite for scale 3.

Word Level Language Modeling

Finally, we evaluate ASRNNs for the word level language modeling (WLLM) task on the WikiText-2 (Merity et al. 2016) dataset, which contains 2M training tokens with a vocabulary size of 33k. We use perplexity as the evaluation metric and the results are summarized in the Table 4, which shows ASRNNs can also outperform their regular

Table 4: Perplexities for word level language modeling on WikiText-2 dataset. Italic numbers are reported by original papers.

METHODS	N	# OF WEIGHTS	MIN. SCALE	MAX. SCALE	AVG. SCALE	PPL
LSTM	1024	$\approx$ 10M	0	0	0	101.1
SLSTM	1024	$\approx$ 10M	3	3	3	97.7
ASLSTM	1024	$\approx$ 10M	0	3	1.51	**93.8**
GRU	1024	$\approx$ 7.8M	0	0	0	99.7
SGRU	1024	$\approx$ 7.8M	3	3	3	95.4
ASGRU	1024	$\approx$ 7.8M	0	3	1.38	**92.6**
ZONEOUT + VARIATIONAL LSTM (MERITY ET AL. 2016)	-	-	-	-	-	100.9
POINTER SENTINEL LSTM (MERITY ET AL. 2016)	-	-	-	-	-	80.8
NEURAL CACHE MODEL (GRAVE, JOULIN, AND USUNIER 2016)	1024	-	-	-	-	**68.9**

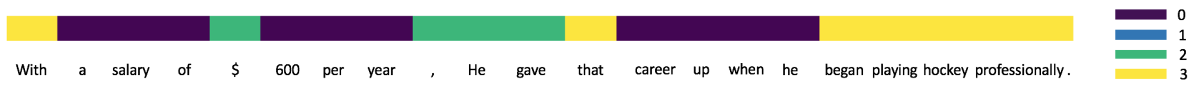


Figure 4: Visualized scale variations for a sampled sentence form WikiText-2 dataset.

counterparts. Besides, Figure 4 further visualizes captured scale variations for a sampled sentence. It indicates scales are usually changed at some special tokens (like semicolon and clause), which comfirms the flexibility of modeling dynamic scale patterns with ASRNNs. What's more, although state-of-the-art models (Merity et al. 2016) (Grave, Joulin, and Usunier 2016) perform better, their techniques are orthogonal to our scaling mechanism so ASRNNs can still benefit from them.

Conclusion

We present Adaptively Scaled Recurrent Neural Networks (ASRNNs), a simple yet useful extension that brings dynamical scale modeling abilities to existing RNN structures. At each time step, ASRNNs model the scale patterns by taking causal convolutions between wavelet kernels and input sequences such that the scale can be represented by wavelet scale coefficients. These coefficients are sampled from Gumbel-Softmax (GM) distributions which are parameterized by previous hidden states and current inputs. The differentiable nature of GM allows ASRNNs to learn to adjust scales based on different temporal contexts. Compared with other multiscale RNN models, ASRNNs don't rely on hierarchical architectures and prefixed scale factors, making them simple and easy to train. Evaluations on various sequence modeling tasks indicate ASRNNs can outperform those non-dynamically scaled baselines by adjusting scales according to different temporal information.

Acknowledgment

This research was partially supported by NSF grant #1704309.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, 265–283.

Arjovsky, M.; Shah, A.; and Bengio, Y. 2016. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, 1120–1128.

Aucouturier, J.-J., and Pachet, F. 2002. Finding songs that sound the same. In *Proc. of IEEE Benelux Workshop on Model based Processing and Coding of Audio*, 1–8.

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Campos, V.; Jou, B.; Giró-i Nieto, X.; Torres, J.; and Chang, S.-F. 2017. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*.

Chang, S.; Qi, G.-J.; Aggarwal, C. C.; Zhou, J.; Wang, M.; and Huang, T. S. 2014. Factorized similarity learning in networks. In *Data Mining (ICDM), 2014 IEEE International Conference on*, 60–69. IEEE.

Chang, S.; Zhang, Y.; Han, W.; Yu, M.; Guo, X.; Tan, W.; Cui, X.; Witbrock, M.; Hasegawa-Johnson, M. A.; and Huang, T. S. 2017. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, 76–86.

Chung, J.; Ahn, S.; and Bengio, Y. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cooijmans, T.; Ballas, N.; Laurent, C.; Gülçehre, Ç.; and Courville, A. 2016. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*.

Defferrard, M.; Benzi, K.; Vandergheynst, P.; and Bresson, X. 2017. Fma: A dataset for music analysis. In *18th International Society for Music Information Retrieval Conference*.

Dieleman, S., and Schrauwen, B. 2014. End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 6964–6968. IEEE.

El Hihi, S., and Bengio, Y. 1996. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, 493–499.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Grave, E.; Joulin, A.; and Usunier, N. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

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

Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J.; et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hu, H., and Qi, G.-J. 2017. State-frequency memory recurrent neural networks. In *International Conference on Machine Learning*, 1568–1577.

Hu, H.; Wang, Z.; Lee, J.-Y.; Lin, Z.; and Qi, G.-J. 2017. Temporal domain neural encoder for video representation learning. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, 2192–2199. IEEE.

Hua, X.-S., and Qi, G.-J. 2008. Online multi-label active annotation: towards large-scale content-based video search. In *Proceedings of the 16th ACM international conference on Multimedia*, 141–150. ACM.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Koutnik, J.; Greff, K.; Gomez, F.; and Schmidhuber, J. 2014. A clockwork rnn. *arXiv preprint arXiv:1402.3511*.

Le, Q. V.; Jaitly, N.; and Hinton, G. E. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Li, K.; Qi, G.-J.; Ye, J.; and Hua, K. A. 2017. Linear subspace ranking hashing for cross-modal retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (9):1825–1838.

Lin, T.; Horne, B. G.; Tino, P.; and Giles, C. L. 1996. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks* 7(6):1329–1338.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

McKay, C., and Fujinaga, I. 2006. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR*, 101–106.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mozer, M. C. 1992. Induction of multiscale temporal structure. In *Advances in neural information processing systems*, 275–282.

Neil, D.; Pfeiffer, M.; and Liu, S.-C. 2016. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, 3882–3890.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 1310–1318.

Qi, G.-J.; Aggarwal, C. C.; and Huang, T. S. 2012. On clustering heterogeneous social media objects with outlier links. In *Proceedings of the fifth ACM international conference on Web search and data mining*, 553–562. ACM.

Qi, G.-J.; Hua, X.-S.; and Zhang, H.-J. 2009. Learning semantic distance from community-tagged media collection. In *Proceedings of the 17th ACM international conference on Multimedia*, 243–252. ACM.

Qi, G.-J. 2016. Hierarchically gated deep networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2267–2275.

Sainath, T. N.; Weiss, R. J.; Senior, A.; Wilson, K. W.; and Vinyals, O. 2015. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Scaringella, N.; Zoia, G.; and Mlynek, D. 2006. Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine* 23(2):133–141.

Schmidhuber, J. 1991. Neural sequence chunkers.

Semeniuta, S.; Severyn, A.; and Barth, E. 2016. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*.

Tang, J.; Hua, X.-S.; Qi, G.-J.; and Wu, X. 2007. Typicality ranking via semi-supervised multiple-instance learning. In *Proceedings of the 15th ACM international conference on Multimedia*, 297–300. ACM.

Tang, J.; Shu, X.; Qi, G.-J.; Li, Z.; Wang, M.; Yan, S.; and Jain, R. 2017. Tri-clustered tensor completion for social-aware image tag refinement. *IEEE transactions on pattern analysis and machine intelligence* 39(8):1662–1674.

Tieleman, T., and Hinton, G. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. Technical report, Technical Report. Available online: https://zh. coursera. org/learn/neuralnetworks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude (accessed on 21 April 2017).

Van Den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Wang, J.; Zhao, Z.; Zhou, J.; Wang, H.; Cui, B.; and Qi, G. 2012. Recommending flickr groups with social topic model. *Information retrieval* 15(3-4):278–295.

Wang, X.; Zhang, T.; Qi, G.-J.; Tang, J.; and Wang, J. 2016. Supervised quantization for similarity search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018–2026.

Wisdom, S.; Powers, T.; Hershey, J.; Le Roux, J.; and Atlas, L. 2016. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, 4880–4888.

Xing, Z.; Pei, J.; and Keogh, E. 2010. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter* 12(1):40–48.

Ye, J.; Hu, H.; Qi, G.-J.; and Hua, K. A. 2017. A temporal order modeling approach to human action recognition from multimodal sensor data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 13(2):14.

Zhang, S.; Wu, Y.; Che, T.; Lin, Z.; Memisevic, R.; Salakhutdinov, R. R.; and Bengio, Y. 2016. Architectural complexity measures of recurrent neural networks. In *Advances in Neural Information Processing Systems*, 1822–1830.