



An Investigation of Recurrent Neural Network Architectures for Statistical Parametric Speech Synthesis

Sivanand Achanta, Tejas Godambe, Suryakanth V Gangashetty

International Institute of Information Technology, Hyderabad, INDIA

{sivanand.a, tejas.godambe}@research.iiit.ac.in, svg@iiit.ac.in

Abstract

In this paper, we investigate two different recurrent neural network (RNN) architectures: Elman RNN and recently proposed clockwork RNN [1] for statistical parametric speech synthesis (SPSS). Of late, deep neural networks are being used for SPSS which involve predicting every frame independent of the previous predictions, and hence requires post-processing for ensuring smooth evolution of speech parameters. RNNs, on the other hand, are intuitively better suited for the task as they inherently model temporal dependencies, but were restricted in use because of the difficulty in training. Lately, techniques such as sparse initialization, Nesterov's accelerated gradient, gradient clipping and leaky integration (LI) have been shown to overcome this difficulty. We study the utility of these techniques for SPSS task. In addition, we show that clockwork RNN is equivalent to an Elman RNN with a particular form of LI. This perspective enables us to understand the reason why a simple Elman RNN with LI units performs well on sequential tasks.

Index Terms: clockwork RNN, statistical speech synthesis, recurrent neural networks, leaky integration

1. Introduction

Statistical parametric speech synthesis (SPSS) has emerged as a powerful technique besides unit selection for text-to-speech conversion. Success of SPSS could be attributed to small foot print system [2] and the flexibility it provides to synthesize expressive voices [3]. However, there are three key issues with SPSS that researchers are trying to tackle [4]. They are as follows: (a) Vocoding: The simple excitation model used in most vocoders results in a buzzy synthesis [5]; (b) Acoustic modeling: The conventional GMM-HMM model has limited ability to model the dependencies across features in a speech frame; (c) Parameter generation: Over smoothing which results because of the way parameters are generated from the acoustic model. In this paper, we describe our experiments with some alternate acoustic models namely RNNs which are aimed at alleviating the above mentioned problem.

Deep neural networks (DNN) have lately received attention from the speech synthesis community [6] [7] as an alternative form of acoustic modeling to traditional HMM-based SPSS. Deep neural networks have a better ability to capture the dependencies across the features, for example, various spectral features and F_0 . However they lack the ability to capture the relations that are spread across time. The text and acoustic features are mapped frame-wise, assuming that each frame is independent of the other [8]. This assumption, although not completely true in case of speech, has been made previously because of difficulty in modeling the temporal correlations with simple models. The output of the deep neural networks consequently is

not smooth from frame to frame and hence had to be explicitly smoothed using maximum output probability parameter generation algorithm [6] [9].

RNNs, on the other hand, are intuitively better suited for the above task as they can model the temporal correlations between successive frames. In this paper we explore two varieties of recurrent neural networks (a) Elman type RNN [10] and (b) the recently proposed clockwork RNN (abbr. CWRNN) [1].

Elman RNNs have long known to be powerful models for tasks such as sequence prediction and generation but their use was restricted because of the difficulty in training them on account of the exploding and vanishing gradients problem [11]. There have been efforts in the recent past toward circumventing this problem primarily in the following directions: (a) Searching for better initialization methods [12]; (b) Adding momentum to speed up learning, and making stochastic gradient descent based back-propagation through time (BPTT) [13], work comparable to a few second order methods (previously proposed to alleviate the problem [14]); (c) Incorporating heuristics such as gradient clipping [11] whenever norm of the gradients crossed a threshold; (d) Using regularization for avoiding vanishing gradients problem [15]. The RNN learning problem cast as a primal dual optimization problem in [16], was shown to avoid gradient explosion and has proved to be effective for speech recognition task on benchmark TIMIT corpus. These studies strongly motivated us to explore RNNs for SPSS by altering the basic BPTT algorithm using the techniques mentioned above.

CWRNN [1] is a new architecture proposed to overcome the learning problem in a different way than mentioned above. It showed that CWRNN, which is a simpler architecture than Elman RNN, is effective for sequence generation and classification tasks, and also surprisingly outperforms Elman RNN and long-short term memory (LSTM) architecture. For this reason we chose to explore CWRNN in addition to Elman RNN for SPSS. In Section 3.2, we give detailed description of this model and its relation to concept of leaky integration.

This paper is organized as follows: Following section gives an account of related work. In Section 3 we describe Elman RNN and clockwork RNN in detail and the heuristics necessary to be incorporated for better learning. In Section 4 we furnish the experimental setup and results of using RNNs in SPSS and compare with DNN as baseline method. Section that follows concludes the paper and possible future directions are given.

2. Relation to Previous Work

A special form of RNNs called LSTMs have been recently applied for SPSS [8] [17]. LSTMs [18] are an alternate way of dealing with vanishing and exploding gradient problems. LSTMs use special kind of memory cells to retain the past

for longer duration and have shown to outperform DNN based models in speech recognition [19] and SPSS [17]. In [8] a unidirectional LSTM was used for low-latency SPSS while its bidirectional counterpart BLSTMs were used in [17]. In [20] a LSTM-RNN was used for directly predicting the waveform instead of intermediate spectral and F_0 features. However, as stated in the introduction, our work is motivated from the observations in [12] [1] [11], which claim to give equally good or better performance than LSTMs with simple RNN architectures.

3. RNNs for SPSS

3.1. Elman RNN

An Elman RNN is a simple RNN with hidden-to-hidden recurrent connections and can be formally described as following,

$$h_t = f(W_i x_t + W h_{t-1} + b_h) \quad (1)$$

$$y_t = g(U h_t + b_o) \quad (2)$$

where W_i represents the input to hidden weights, W the recurrent weights of hidden layer and b_h the hidden biases, U the hidden to output weights and b_o the biases of output neurons. f, g are the nonlinear functions at hidden and output layers respectively. x_t, h_t, y_t are input, state and output at time t respectively and h_{t-1} is the state at previous time instant $t - 1$. In our case, since it was a regression task output layer was kept linear and \tanh units were used in hidden layer. All the equations given below are for g being identity function.

The parameters of the output layer can be learned using normal back-propagation, and other parameters of the model are learned using back-propagation through time. The recursion for computing the error signal at hidden layer through BPTT is given as [13]

$$\begin{aligned} e_t &= y_t - d_t \\ \delta_t &= f' * (W^T \delta_{t+1} + U^T e_t) \end{aligned} \quad (3)$$

e_t, δ_t represent error signal at time t at output layer and hidden layer respectively. However, the naive implementation of BPTT will cause gradient explosion or vanishing phenomenon. In the following subsections we discuss few tricks suggested in the literature [15] [12] for better training.

3.1.1. Sparse Initialization

A new type of initialization scheme called ‘‘sparse initialization’’ was introduced in [21]. In this scheme, each random unit is connected to 15 randomly chosen units from the previous layer. This provides following advantages: (a) The fan-in to a neuron does not depend on the size of previous layer and as a result the units do not easily saturate. (b) Each unit receives inputs from different set of neurons and hence they tend to be qualitatively more diverse. We used this initialization scheme only for recurrent weights (W) while other weights were densely initialized as usual. The sparsely initialized W was re-scaled to make sure that the spectral radius was around 1.1. All the weights were drawn from zero mean Gaussian distribution $\sigma N(0, 1)$, scaled by a constant σ , with σ assuming values one among 0.1, 0.01, 0.001. It was suggested in [12] that scale of the Normal distribution plays a significant role and especially is problem dependent for W_i weights.

3.1.2. Nesterov’s Accelerated Gradient

Nesterov’s accelerated gradient was shown to perform better than classical momentum in [22]. The following equation implements the Nesterov’s momentum.

$$\begin{aligned} v_t &= \mu v_{t-1} - \eta \nabla J(\theta_t + \mu v_{t-1}) \\ \theta_t &= \theta_{t-1} + v_t \end{aligned} \quad (4)$$

where J, v, θ represent cost function (mean squared error in our case), velocity and parameter set respectively.

3.1.3. Gradient Clipping

Gradient clipping was introduced in [15] to avoid the gradient explosion problem. In gradient clipping, average length of gradients is computed over one pass of the training data and a scaled version of this average length is used as threshold (th). If the length of the gradient vector exceeds this value, then gradients are re-scaled to have length equal to th .

$$g = \frac{th}{\|g\|} \times g \quad (5)$$

where g represents the gradient vector whose threshold is greater than th . This is done only to input and recurrent weight gradients.

3.1.4. Leaky Integration

In [15] a low-pass filtering mechanism is used to overcome the vanishing gradients problem. Specifically, Eq 1, is modified as a convex combination of current activation and previous hidden state. Note that scaling constants (α_i) vary from neuron to neuron with i being the hidden neuron index. In our study we vary the scaling constants from 0.02 to 0.2 for some neurons while keeping it 0 for the rest. If $\alpha = 0$ the model boils down to the usual RNN. The new forward pass equation is given by

$$h_{t,i} = \alpha_i h_{t-1,i} + (1 - \alpha_i) f_i(W_i x_t + W h_{t-1} + b_h) \quad (6)$$

3.2. Clockwork RNN

CWRNN [1] is a new variant of RNN in which neurons in hidden layer are split into several groups and update of each group is dependent on the time step. Fig. 1 shows the architecture of CWRNN taken from [1]. Each group of neurons is fully connected within but connections across groups are restricted. Neurons in one group are connected to other group only if it is relatively faster i.e. $T_i < T_j$ (see Fig. 1). Because of the restricted connections the recurrent weight matrix is a block-upper triangular matrix. It is an asynchronous architecture where only few neurons are active at a given time step. A temporal schedule is designed according to which groups are made active, for ex. $T = 1, 2, 4, 8, 16, 32$ is an exponential schedule that is used in our experiments based on which neurons in the hidden layer are split into 6 groups. Group i becomes active if current step is a multiple of T_i implying that only those rows in W_i, W and b_h are active. The activation values of neurons in inactive groups at a particular time step are copied from previous time step. A more detailed description of forward pass of CWRNN can be found in [1].

3.2.1. A Leaky Integration Interpretation

CWRNN model can be interpreted as an Elman RNN with time-varying LI units having specific form of α_i as briefed below. In

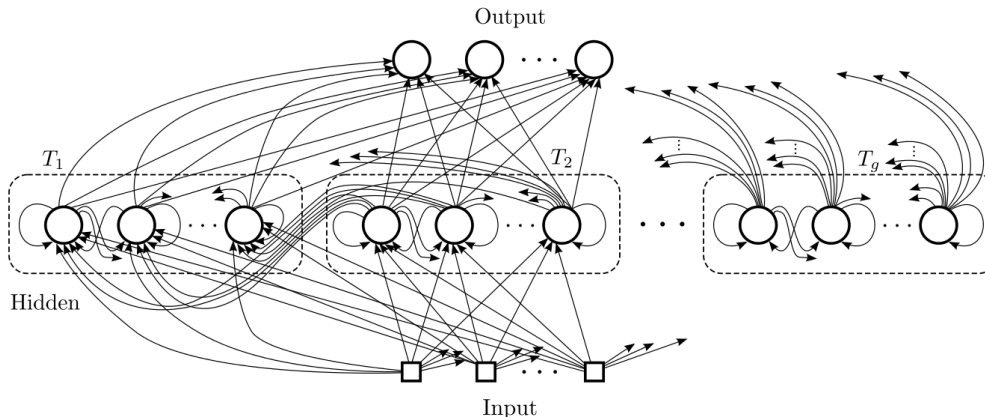


Figure 1: Clockwork RNN architecture

equation 6 if some α_i s are made equal to one (for those neurons which are inactive at a given time step t) and the rest equal to zero, this process is time varying instead of static, and resembles the CWRNN forward pass.

$$h_{t,i} = \alpha_{t,i}h_{t-1,i} + (1 - \alpha_{t,i})f_i(W_ix_t + Wh_{t-1} + b_h) \quad (7)$$

This key insights lets us derive the BPTT algorithm for training CWRNN as

$$\begin{aligned} e_t &= y_t - d_t \\ \delta_t &= (1 - \alpha_t)f' * (W^T\delta_{t+1} + U^Te_t) + \alpha_t\delta_{t+1} \end{aligned} \quad (8)$$

Because the time varying α_i with each of its element being binary has been shown to work well in [1], we have decided to experiment with the α as mentioned in Section 3.1.4, where only the time varying part was removed and a continuum of values from 0.02 to 0.2 instead of 0 or 1 is taken.

4. Experiments and Results

Using the above models and DNN as our baseline method, we built several SPSS systems. Our experiments were done using BDL speaker ARCTIC database [23]. The code for replicating the experiments and samples are available at ¹. All our experiments were run on NVIDIA Geforce GTX-660 graphics card. We have extracted the full context labels from labelled data. This includes quinphone identities along with vowel in the current syllable as categorical features. The numerical features include the number of phones in the syllable, number of syllables in the word, number of words in the utterance and so on. The total dimension of the input feature vector was 305. For RNNs, the previous and previous to previous phone contexts were not included as the past is memorized by the model as suggested in [8]. The phone level duration and frame indicators were included as duration features at input. Input features were mean and variance normalized. At the output we just had spectrum extracted from the wave files using STRAIGHT tool (dimensionality was 513). The frame shift was set to 5ms. The

¹<https://github.com/SivanandAchanta/RNN>

output features were left un-normalized (we found that normalizing between 0.01 and 0.99 did not improve the performance).

Although we could have in principle included F_0 and aperiodicity we have used their natural versions during synthesis as our main aim was to investigate how RNNs perform rather than building a complete synthesizer. There were total 1131 sentences of which 913 were used for training, 100 for validation and remaining for testing. The best hyper-parameters were obtained by fine tuning on validation set.

A deep neural network with architecture 305L 600R 600R 513L, i.e., 2 hidden layers with 600 neurons in each and rectified linear units (R) has been trained using a minibatch stochastic gradient descent with adadelata [24] for learning rate setting and Nesterov's accelerated gradient based momentum [12]. A GPU-based mini-batch stochastic gradient descent (M-SGD) was implemented in MATLAB. The momentum factor was set to 0.9 after fine-tuning on validation dataset. The mini-batch size was set to 530 so that the number of updates per epoch will be approximately equal to recurrent neural network approach based updates.

For training various versions of RNNs as given below, each utterance was treated as a sequence and for one full epoch the trainer was presented with 913 sequences. It took approximately 1 day to train each RNN for 20 epochs. All the RNNs were trained with 600 hidden units. Note that gradient clipping was also used in CWRNN although it was not mentioned in [1]. Learning rate was kept fixed at 0.0001 and momentum factor of 0.9 for first 1000 updates and 0.98 till the end of optimization. Surprisingly, momentum factor choice did not greatly influence the performance contrary to what was reported in [12]. This possibly could be because, we use a purely stochastic gradient descent approach, wherein the gradients are noisy. A similar observation was also made in a recent study [25].

The plot of test set error per weight update is given in Fig. 2 from which we can see the effect of various modification schemes described in Section 3.1. Baseline DNN is shown with magenta dash-dotted line. RNN with dense initialization (blue line) can be regarded as the vanilla RNN or the plain BPTT which performs worse than DNN. Dense initialization with spectral radius re-scaled to 1.1 (black line) also showed no improvement and coincides with blue line. The average lengths of the gradients with dense initialization over one epoch was in the order of 0.01 and hence leading to vanishing gradients problem. Next we experimented with LI (red line) in order to

overcome the problem of vanishing gradients and we can see that it indeed leads to a slightly better local minimum although still is poorer than DNN.

Sparse initialization (red dash-dotted line) alone leads to a better optimum than RNN with dense initialization and LI, reinforcing the fact that initialization is indeed important for better optimization [12]. However, it is only sparse initialization coupled with gradient clipping (black dash-dotted line) that leads to a minimum that is better than DNN. Using LI (magenta line) units only improved slightly.

Clockwork RNN (blue dash-dotted line), although has block-upper triangular recurrent weight matrix, it is still dense and hence it can be expected to perform closely to RNN with dense initialization and LI which indeed is the case.

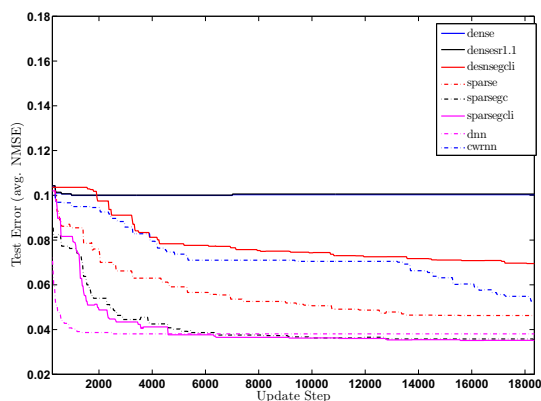


Figure 2: Test set error Vs. Update step (best viewed in color)

We report the average normalized mean squared error (ANMSE) obtained on test set at the last epoch of training DNN, RNN, RNN with gradient clipping (RNN-C), RNN with clipping and leaky integration (RNN-CL) and finally the clockwork RNN in Table 1. ANMSE of RNN with both initialization schemes are shown. Table 1 clearly indicates that RNN with sparse initialization, gradient clipping and LI is able to outperform DNN.

Table 1: Average normalized mean squared error on test set at final epoch

Init	DNN	RNN	RNN-C	RNN-CL	CWRNN
Dense	0.0381	0.099	-	0.0695	0.0527
Sparse	-	0.0463	0.0353	0.0351	-

5. Conclusions and Future Work

In this work, we experimented with two recurrent neural network architectures for SPSS task. The preliminary experiments conducted led to some insights on how to train simple Elman RNN for achieving better results than DNN. Our experiments confirm that due to the vanishing gradients problem densely initialized vanilla RNNs are difficult to optimize. Sparse initialization helps us to overcome that problem while the gradient explosion problem cropped up. This problem was then solved by using gradient clipping. LI units do not show significant impact because the combined effect of sparse initialization and gradient clipping are able to take care of the optimization of RNNs.

We also see that momentum factor does not play a significant role which comes as a surprise to us. The CWRNN architecture was shown to be equivalent to RNN with LI units and it performs better than RNN with dense initialization and LI units. In future we wish to experiment CWRNN with sparse initialization. Also, many more investigations are warranted especially using more amount of data. Future work is being carried out in two directions, one in predicting duration and F_0 in addition to spectrum from RNNs using audiobook data and the other in using multiplicative RNNs or Tensor RNNs for improving the acoustic modeling in SPSS.

6. Acknowledgements

The first author would like to thank Klaus Greff, Heiga Zen for clarifying some concepts related to clockwork RNN and DNN based SPSS respectively. Thanks are also to Prof. Hideki Kawahara for making the STRAIGHTV40 code available for use. Authors would like to thank Dr. Kishore Prahallad for helpful discussions and Sai Krishna R for proofreading the paper.

7. References

- [1] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A Clockwork RNN," in *Proc. of ICML*, 2014, pp. 1863–1871.
- [2] A. Gutkin, X. Gonzalvo, S. Breuer, and P. Taylor, "Quantized HMMs for low footprint text-to-speech synthesis," in *Proc. of Interspeech*, 2010, pp. 837–840.
- [3] J. Yamagishi, T. Masuko, and T. Kobayashi, "A style control technique for HMM-based expressive speech synthesis," *IEICE Transactions on Information and Systems*, vol. 90, no. 9, pp. 1406–1413, 2007.
- [4] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039 – 1064, 2009.
- [5] S. Imai, "Cepstral analysis synthesis on the mel frequency scale," in *Proc. of ICASSP*, 1983, pp. 93–96.
- [6] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. of ICASSP*, 2013, pp. 7962–7966.
- [7] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of Deep Neural Network (DNN) for parametric TTS synthesis," in *Proc. of ICASSP*, 2014, pp. 3829–3833.
- [8] H. Zen and H. Sak, "Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis," in *Proc. of ICASSP*, 2015.
- [9] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proc. of ICASSP*, 2000, pp. 1315–1318.
- [10] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.
- [11] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. of ICML*, 2013, pp. 1310–1318.
- [12] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. of ICML*, 2013, pp. 1139–1147.
- [13] R. J. Williams and J. Peng, "An Efficient Gradient-Based Algorithm for On-line Training of Recurrent Network Trajectories," *Neural Computation*, vol. 2, pp. 490–501, 1990.
- [14] J. Martens, "Deep learning via hessian-free optimization," in *Proc. of ICML*, 2010, pp. 735–742.
- [15] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *Proc. of ICASSP*, 2013, pp. 8624–8628.

- [16] J. Chen and L. Deng, "A primal-dual method for training recurrent neural networks constrained by the echo-state property," *arXiv preprint arXiv:1311.6091*, 2013.
- [17] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS Synthesis with Bidirectional LSTM Based Recurrent Neural Networks," in *Proc. of Interspeech*, 2014.
- [18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [19] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.
- [20] K. Tokuda and H. Zen, "Directly Modeling Speech Waveforms by Neural Networks for Statistical Parametric Speech Synthesis," in *Proc. of ICASSP*, 2015.
- [21] J. Martens and I. Sutskever, "Learning Recurrent Neural networks with Hessian-free Optimization," in *Proc. of ICML*, 2011, pp. 1033–1040.
- [22] I. Sutskever, "Training recurrent neural networks," Ph.D. dissertation, University of Toronto, 2013.
- [23] J. Kominek and A. W. Black, "The CMU ARCTIC speech databases," in *Proc. of 5th ISCA Speech Synthesis Workshop*, 2004, pp. 223–224.
- [24] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.
- [25] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.