SPEECH RECOGNITION WITH PREDICTION-ADAPTATION-CORRECTION RECURRENT NEURAL NETWORKS

Yu Zhang *

MIT CSAIL
Cambridge, MA, USA
yzhang87@csail.mit.edu

Dong Yu, Michael L. Seltzer, Jasha Droppo

Microsoft Research
One Microsoft Way, Redmond, WA, USA
{dongyu,mseltzer,jdroppo}@microsoft.com

ABSTRACT

We propose the prediction-adaptation-correction RNN (PAC-RNN), in which a correction DNN estimates the state posterior probability based on both the current frame and the prediction made on the past frames by a prediction DNN. The result from the main DNN is fed back to the prediction DNN to make better predictions for the future frames. In the PAC-RNN, we can consider that, given the new, current frame information, the main DNN makes a correction on the prediction made by the prediction DNN. Alternatively, it can be viewed as adapting the main DNN's behavior based on the prediction DNN's prediction. Experiments on the TIMIT phone recognition task indicate that the PAC-RNN outperforms DNN, RNN, and LSTM with 2.4%, 2.1%, and 1.9% absolute phone accuracy improvement, respectively. We found that incorporating the prediction objective and including the recurrent loop are both important to boost the performance of the PAC-RNN.

Index Terms— Deep Neural Network, DNN, Recurrent neural network, RNN, Prediction-Adaptation-Correction RNN, PAC-RNN

1. INTRODUCTION

The deep neural network (DNN)-based acoustic models (AMs) have greatly improved automatic speech recognition (ASR) accuracy on many tasks [1, 2, 3, 4]. Very recently, further improvements were reported by using more advanced models such as convolutional neural networks (CNNs) [5, 6, 7, 8] and long short-term memory (LSTM) recurrent neural networks (RNNs) [9, 10, 11]. These advancements have reduced the word error rate (WER) to below 10% in many real world applications with close-talk microphones.

Despite the great progress, the ASR performance under noisy reverberant conditions and on multi-talker speech are still far from satisfactory [12, 13]. The effort to seek a more powerful AM continues. In this paper we propose the prediction-adaptation-correction RNN (PAC-RNN), in which a main (or correction) DNN estimates the state posterior probability based on both the current frame information and the prediction made on the past frames by a prediction DNN. The result from the main DNN is fed back to the prediction DNN to make better predictions for the future frames. In this model, we can consider that, given the new, current frame information, the main DNN makes a correction on the prediction made by the prediction DNN. Alternatively, it can be considered that the main DNN's behavior is adapted based on the prediction made by the prediction DNN. Although the concept of prediction-adaptation-correction is not new and arises naturally from Kalman filters [14] for example, our specific architecture and its application in ASR are novel.

The behavior of prediction, adaptation, and correction is widely observed in human speech recognition. For example, listeners may guess what you will say next and wait to confirm their guess. They may adjust their listening effort by predicting the speaking rate and noise condition based on the current information, or predict and adjust the mapping from letter to sound based on the speaker's current pronunciation. They may even predict what your next sound will be and focus their attention to only the relevant part in the audio signal. Although the prediction and adaptation ability of the PAC-RNN described in this paper is much more limited compared to that of humans, we believe it is a valuable first step toward the right direction.

We evaluated the PAC-RNN on the TIMIT phone recognition task. Compared to the DNN, RNN, and LSTM, the PAC-RNN achieved 2.4%, 2.1%, and 1.9% absolute phone accuracy improvement, respectively. We investigated the effects of choosing different prediction targets, using different contextual window, including and removing the recurrent connection, and including and excluding the prediction criterion in the training objective function in the PAC-RNN. We found that to achieve the best result the PAC-RNN should predict targets of a distant future, incorporate the prediction objective in the training criterion, and include the recurrent loop to exploit long-range dependency.

The rest of the paper is organized as follows. We describe the PAC-RNN in detail in Section 2 and evaluate its performance against different configurations in Section 3. Related work is discussed in Section 4. We conclude the paper in Section 5.

2. PREDICTION-ADAPTATION-CORRECTION RECURRENT NEURAL NETWORKS

2.1. Model Structure

Figure 1 illustrates the structure of the PAC-RNN studied in this paper. At the center of the model is a main (or correction) DNN and a prediction DNN. The main DNN estimates the state posterior probability $p^{corr}(s_t|\mathbf{o}_t, \mathbf{x}_t)$ given $\mathbf{o}_t$, the observation feature vector, and $\mathbf{x}_t$, the information from the prediction DNN, at time t. The prediction DNN predicts some target information in the future. In this study, it predicts the posterior probability $p^{pred}(l_{t+n}|\mathbf{o}_t, \mathbf{y}_t)$ given $\mathbf{o}_t$ and $\mathbf{y}_t$, the information from the correction DNN, where l can be a state s or a phone θ, and n is the number of frames look ahead. Note that since $\mathbf{y}_t$, the information from the correction DNN, depends on $\mathbf{x}_t$, the information from the prediction DNN, and vice versa, a recurrent loop is formed.

Here the information from the prediction and correction DNNs

*This work was done while Yu Zhang was an intern at Microsoft Research.

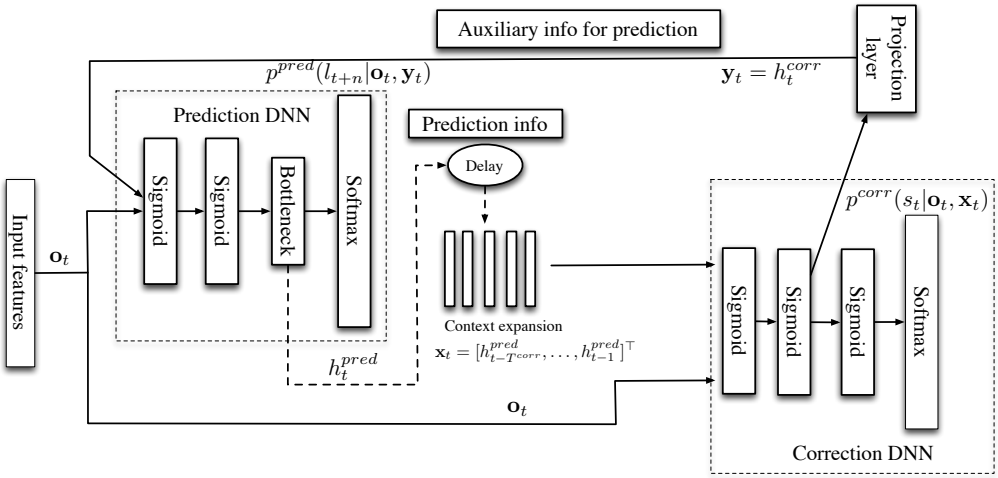

Fig. 1. The Structure of the PAC-RNN

can be drawn from either the softmax layer or a hidden layer. In large vocabulary speech recognition (LVSR) tasks there are often over 5000 states. In these cases, drawing information from the softmax layer can significantly increase the model size. For this reason, we obtained information from a (bottleneck) hidden layer whose size can be set independent of the state size so that the same architecture can be applied to the LVSR tasks directly.

In a very basic setup, $\mathbf{x}_t$, the information from the prediction DNN, is simply the bottleneck hidden layer output value h_{t-1}^{pred}. To exploit additional predictions made in the past, however, we can stack multiple hidden layer values as

$$\mathbf{x}_t = [h_{t-T^{corr}}^{pred}, ..., h_{t-1}^{pred}]^T, \quad (1)$$

where T^{corr} is the contextual window size used by the correction DNN and is set to 10 in our study. Similarly, we can stack multiple frames to form $\mathbf{y}_t$, the information from the correction DNN, as

$$\mathbf{y}_t = [h_{t-T^{pred}-1}^{corr}, ..., h_t^{corr}]^T, \quad (2)$$

where T^{pred} is the contextual window size used by the prediction DNN and is set to 1 in our study. In addition, in the specific example shown in Figure 1, the hidden layer output h_t^{corr} is projected to a lower dimension before it is fed into the prediction DNN.

2.2. Training and Decoding

To train the PAC-RNN, we need to provide supervision information to both the prediction and correction DNNs. As we have mentioned, the correction DNN estimates the state posterior probability, and thus the state label and the frame cross-entropy (CE) criterion can be used. For the prediction DNN, however, we have freedom to choose either the state or the phoneme label. We will compare the performance difference between these two choices in Section 3.

The PAC-RNN training problem is a multi-task learning problem. The two training objectives can be combined into a single one as

$$J = \sum_{t=1}^{T} (\alpha * \ln p^{corr}(s_t|\mathbf{o}_t, \mathbf{x}_t) + (1-\alpha) * \ln p^{pred}(l_{t+n}|\mathbf{o}_t, \mathbf{y}_t)), \quad (3)$$

where α is the interpolation weight and is set to 0.8 in our study unless otherwise stated, and T is the total number of frames in the training utterance.

During the decoding stage, the state posteriors (or the scaled likelihood scores converted from them) from the correction DNN are treated as the emission probability similar to that in a typical DNN/RNN-HMM hybrid system [3, 15].

2.3. Implementation

We implemented the PAC-RNN using the computational network toolkit (CNTK) [16]. CNTK can train RNNs with arbitrary recurrent connections. We only need to prepare the training data and a text description of the model. CNTK uses a delay operation to retrieve values in the past. Thus, hidden layer values such as h_{t-k}^{pred} can be easily described as delaying the value h_t^{pred} for k frames.

In this study, the truncated back-propagation-through-time (BPTT) [17] is used to update the model parameters and each utterance is truncated into multiple segments. At the beginning of each utterance a default value is assigned to the input of the delay node. The delay node in the later segments can retrieve values from previous segments. In other words, the forward computation is carried out across segments. To speed up the training, we process multiple utterances simultaneously as a batch. We have found that this not only reduces training time but also improves the quality of the final model. In this study each BPTT segment contains 20 frames and we process 5 utterances simultaneously.

3. EXPERIMENTS

In this section, we evaluate PAC-RNN on the TIMIT phone recognition task. In our experiments the training labels are obtained through forced alignment using our GMM-HMM system trained with the maximum-likelihood criterion [1]. The standard 462-speaker training set is used, and all SA sentences are removed in order to conform to the standard setup as in [18]. A separate development set of 50 speakers is used for tuning all hyper parameters. Results are reported

[1] In [18], the expertly-annotated phone boundaries were used to generate the training labels, which outperform the HMM generated labels. We use the HMM generated labels since this is the only label available in other datasets.

on the 24-speaker core test set, which has no overlap with the development set.

3.1. Results Summary

Table 1 summarizes the phone accuracy achieved with different hybrid models evaluated in this study. All the DNN/RNN models use a 123 dimensional acoustic feature vector, consisted of 40 dimensional mel-frequency log-filterbank features, an energy measure, and their first and second temporal derivatives. In our experiments, 183 target class labels, corresponding to three states for each of 61 phones, are used. A bi-gram phone language model estimated from the training set is used in decoding. The language model weight is tuned on the development set.

We consider three baseline hybrid systems: a DNN with two 2048-unit hidden layers, a simple RNN with two 2048-unit hidden layers in which the final hidden layer is a recurrent layer, and an LSTM with 1024 memory cells. We don't see further performance improvement by increasing the model sizes of these baseline systems.

In the PAC-RNN (S) model, the prediction DNN has a 1024-unit hidden layer and a 80-unit bottleneck layer. The correction DNN has two 1024-unit hidden layers. The projection layer from the correction DNN's hidden layer contains 500 neurons. In the PAC-RNN (L) model, all 1024-unit hidden layers are replaced with 2048-unit hidden layers.

For the DNN, simple RNN, and PAC-RNN models, the input contains a 7-1-7-frame contextual window which translates to a total size of $123 * 15 = 1845$. No context expansion is used for the LSTM model since the best performance is obtained without any context expansion.

All models are randomly initialized without either generative or discriminative pretraining [19]. No momentum is used for the first epoch and a momentum of 0.9 is used for all the subsequent epochs. We have found that turning off the momentum for the first epoch helps to improve the performance of the final model although the model after the first epoch seems to be worse. We believe this is because the randomly initialized model is highly non-optimized and so noisier gradient helps to move the model to a better starting point. To train the DNN, a learning rate of 0.1 per minibatch is used for the first epoch. The learning rate is increased to 1.0 at the second epoch, after which it is kept the same until the development set training criterion no longer improves, under which condition the learning rate is halved. A similar schedule is used to train the RNNs except that all the learning rates are reduced to $1/10$ of that used in the DNN training. Following [18], the state posteriors are directly used as the emission probability in the HMM without first being converted to the scaled likelihood, although converting to scaled likelihood is preferred for LVSR tasks.

The simple RNN only slightly outperforms the DNN which is consistent with other reported results [15]. The LSTM further improves upon the simple RNN. The PAC-RNN (L) outperforms DNN, RNN and LSTM with 2.4%, 2.1%, and 1.9% absolute phone accuracy improvement, respectively, on the core test set.

3.2. Effect of Expanding Prediction Information

As described in Section 2, the prediction information fed into the correction DNN can include multiple past predictions. Table 2 compares the phone recognition accuracy with and without prediction information expansion. From the table, we can observe that if the prediction DNN only predicts the state of the next frame, no gain over the baseline DNN is observed. This is because most frames have the same label as the next frame and so the prediction DNN does not provide much information to the correction DNN. If the prediction DNN predicts the state of the $t + 10$-th frame, however, we can observe a 1.0% phone accuracy improvement over the DNN baseline. An additional improvement of 0.7% is obtained if 10 past predictions are used by the correction DNN. This indicates that the contextual expansion of the prediction information can be very helpful.

Model	Dev	Test	# of Parameters
DNN	79.6%	77.8%	8.4M
Simple RNN	79.5%	78.1%	12.5M
LSTM	79.9%	78.3%	5.90M
PAC-RNN (S)	81.1%	80.0%	6.9M
PAC-RNN (L)	81.6%	80.2%	15.1M

Table 1. TIMIT Phone Accuracy Achieved with Different Hybrid Models

Model	Target	Context Expansion	Dev	Test
DNN	-	-	79.6%	77.8%
PAC-RNN (S)	s_{t+1}	no	79.7%	77.7%
PAC-RNN (S)	s_{t+10}	no	80.3%	78.8%
PAC-RNN (S)	s_{t+10}	yes	80.8%	79.5%

Table 2. Effect of Prediction Information Expansion

3.3. Effect of Different Prediction Targets

In order to determine the best prediction target we compared the PAC-RNN with different prediction targets, all with 10-frame contextual expansion, in Table 3. From the table we can see that predicting a longer future (e.g., next phone) is better than predicting a shorter future (e.g., next state), and predicting a more meaningful unit (e.g., using the next phone as the target) is better than predicting the states over a fixed window size (e.g., the state of the $t + 10$-th frame). The best result is obtained by predicting the next phone. The PAC-RNN (S) outperforms the DNN with 2.2% accuracy improvement while PAC-RNN (L) introduces an additional 0.2% improvement on the core test set.

Model	Target	Dev	Test
DNN		79.6%	77.8%
PAC-RNN (S)	Next state symbol	80.2%	78.6%
PAC-RNN (S)	State of the $t + 10$-th frame	80.8%	79.5%
PAC-RNN (S)	Next phoneme symbol	81.1%	80.0%
PAC-RNN (L)	Next phoneme symbol	81.6%	80.2%

Table 3. Effect of Different Prediction Targets. All with 10 Frames of Expansion.

3.4. Effect of the Recurrent Loop

In this subsection we investigate the effect of the recurrent loop in the PAC-RNN. In Table 4, the setup with the recurrent loop is the PAC-RNN we have described in Section 2. In the setup with no recurrent loop, the connection from the correction DNN back to the prediction DNN is removed while the prediction DNN is still used

to provide prediction information to the correction DNN. From the table, we can observe that including the recurrent loop is critical and can achieve 1.8% accuracy improvement over the system with no recurrent loop, which is only 0.4% better than DNN. We believe this is because long-range information can be exploited more effectively with recurrent connections.

Model	With Recurrent Loop	Dev	Test
DNN		79.6%	77.8%
PAC-DNN (S)	No	80.0%	78.2%
PAC-RNN (S)	Yes	81.1%	80.0%

Table 4. Effect of the Recurrent Loop

3.5. Effect of Optimizing the Prediction Criterion

To train the PAC-RNN we optimize a combined objective function that is an interpolation of the correction and prediction criteria as shown in Eq. 3. By adjusting the interpolation weight α we can change the relative importance of each criterion. Table 5 summarizes the phone recognition accuracy achieved when the PAC-RNN is trained with different interpolation weights. As expected, if we set α to 1.0 to remove the prediction criterion from the training objective function, the PAC-RNN performs almost as well as to the simple RNN and LSTM. If α is set to a small value (e.g., 0.6 in the table) such that the main criterion is not sufficiently emphasized, the performance also degrades but can still be better than the PAC-RNN trained without the prediction criterion.

Model	Interpolation Weight	Dev	Test
DNN		79.6%	77.8%
LSTM		79.9%	78.3%
PAC-RNN (S)	1.0 (no prediction)	80.1%	78.4%
PAC-RNN (S)	0.8	81.1%	80.0%
PAC-RNN (S)	0.6	80.3%	79.0%

Table 5. Phone Recognition Accuracy Achieved with Different Interpolation Weights

4. RELATED WORK

The core concept of prediction, adaptation, and correction has been widely used in models such as Kalman filter [14]. Through multi-pass decoding traditional ASR systems also implicitly exploit the prediction information.

The work that is most similar to ours is that presented in [20]. Their model explicitly predicts multi-frame state labels which are exploited during the decoding time through autoregressive product. Their work, however, does not involve recurrent feedbacks and thus cannot effectively exploit long-range dependencies in the signal. More importantly, their model is very different from ours in spirit and was proposed from a completely different angle.

RNNs can be naturally used to do prediction. However, both the simple RNN [15, 21] and the LSTM [11, 9] RNN that have been successfully applied to AMs do not explicitly make future predictions. In this study, we have included results from using the simple RNN and the LSTM and showed that the PAC-RNN can do better by modeling prediction information more explicitly.

5. CONCLUSION

We perceive that the next-generation ASR systems can be solely described as a dynamic system that involves many connected components and recurrent feedbacks and constantly makes predictions, corrections, and adaptations. For example, the system should be able to automatically identify multiple talkers in the mixed speech and then focus on a specific speaker by ignoring other speakers and noises.

The PAC-RNN we proposed in this paper is a first step towards this direction. It has some of the properties we described above. Although it only makes simple predictions and corrections, it already shows promising potential as indicated by 2.4%, 2.1%, and 1.9% accuracy improvement over the DNN, RNN, and LSTM, respectvely, on the TIMIT phone recognition task. Our future work includes applying the PAC-RNN to tasks on which the conventional models do not work well and extending it by predicting additional information such as the speech signal, speaker, speaking rate, and noise.

6. REFERENCES

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2011, pp. 437–440.

[3] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, , no. 6, pp. 82–97, 2012.

[4] M. Seltzer, D. Yu, and Y. Q. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[5] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8614–8618.

[6] T. N. Sainath, B. Kingsbury, A. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. IEEE Workshop on Automfatic Speech Recognition and Understanding (ASRU)*, 2013, pp. 315–320.

[7] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, 2014.

[8] L. Toth, "Convolutional deep maxout networks for phone recognition," in *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2014.

[9] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[10] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. IEEE Workshop*

on Automfatic Speech Recognition and Understanding (ASRU), 2013, pp. 273–278.

[11] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[12] Y. Huang, D. Yu, C. J. Liu, and Y. F. Gong, "A comparative analytic study on the Gaussian mixture and context dependent deep neural network hidden Markov models," in *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2014.

[13] C. Weng, D. Yu, M. Seltzer, and J. Droppo, "Single-channel mixed speech recognition using deep neural networks," in *Proc. ICASSP*, 2014.

[14] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[15] C. Weng, D. Yu, S. Watanabe, and F. Juang, "Recurrent deep neural networks for robust speech recognition," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5569–5573.

[16] D. Yu, A. Eversole, M. Seltzer, K. Yao, B. Guenter, O. Kuchaiev, F. Seide, H. Wang, J. Droppo, Z. Huang, Y. Zhang, G. Zweig, C. Rossbach, J. Currey, J. Gao, A. May, A. Stolcke, and M. Slaney, "An introduction to computational networks and the computational network toolkit," Tech. Rep. MSR, Microsoft Research, 2014, http://cntk.codeplex.com.

[17] R. Williams and J. Peng, "An efficient gradient-based algorithm for online training of recurrent network trajectories," *Neural Computation*, vol. 2, pp. 490501, 1990.

[18] A. Mohamed, G.E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 14 –22, Jan. 2012.

[19] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE Workshop on Automfatic Speech Recognition and Understanding (ASRU)*, 2011, pp. 24–29.

[20] N. Jaitly, V. Vanhoucke, and G. Hinton, "Autoregressive product of multi-frame predictions can improve the accuracy of hybrid models," in *Proc. Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2014.

[21] L. Deng and J. S. Chen, "Sequence classification using the high-level features extracted from deep neural networks," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014.