REVISITING RECURRENT NEURAL NETWORKS FOR ROBUST ASR

Oriol Vinyals[1,2]*, Suman V. Ravuri*[1,2]*, Daniel Povey*[3]

[1]International Computer Science Institute, Berkeley, CA, USA
[2]EECS Department, University of California - Berkeley, Berkeley, CA, USA
[3]Microsoft Research, Redmond, WA, USA

vinyals@icsi.berkeley.edu, ravuri@icsi.berkeley.edu, dpovey@microsoft.com

ABSTRACT

In this paper, we show how new training principles and optimization techniques for neural networks can be used for different network structures. In particular, we revisit the Recurrent Neural Network (RNN), which explicitly models the Markovian dynamics of a set of observations through a non-linear function with a much larger hidden state space than traditional sequence models such as an HMM. We apply pretraining principles used for Deep Neural Networks (DNNs) and second-order optimization techniques to train an RNN. Moreover, we explore its application in the Aurora2 speech recognition task under mismatched noise conditions using a Tandem approach. We observe top performance on clean speech, and under high noise conditions, compared to multi-layer perceptrons (MLPs) and DNNs, with the added benefit of being a "deeper" model than an MLP but more compact than a DNN.

Index Terms: Automatic Speech Recognition, Recurrent Neural Networks, Deep Learning

1. INTRODUCTION

Using features other than MFCCs has long been a focus of research in the speech recognition community, and the combination of various feature streams has proven useful in a variety of speech recognition systems. A common technique to merge streams is to use a Tandem method [1], in which processed phone posterior probabilities are appended to standard MFCCs.

Typically, these posteriors are generated through some discriminative process, and MLPs have successfully been used in speech for many years. More recently, research on the training strategies for deep networks has allowed for improved performance in many machine learning and pattern recognition tasks, as described in Section 2. Recent efforts in the speech community have explored and expanded the principles of deep learning, and found that deep architectures are more efficient at modeling speech, and generalize better [2, 3, 4]. This model has performed well in a number of different configurations (such as the Hybrid system, in which posterior estimates are transformed to likelihoods and directly used in an HMM [3], or as the Tandem system used in this work).

The typical application of deep learning to speech recognition uses a frame with context (typically of a few frames) as the input of the network, and learns (with or without pretraining) a DNN to estimate frame level phone or state posteriors. Weights are usually updated using backpropagation with stochastic gradient descent. In this paper, we propose a new point of view to the deep learning paradigm:

- Given advances in understanding deep architectures, we pose RNNs as an instantiation of these models, and re-explore previous work on the subject [5], comparing traditional MLPs, DNNs[1], and RNNs under noise environments.

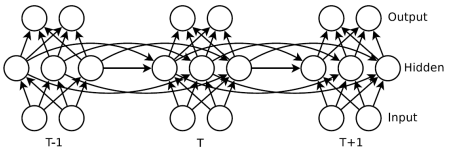

Fig. 1. *Structure of our Recurrent Neural Network.*

The RNN (see Figure 1) is a natural extension to DNNs for temporal sequence data such as speech. In RNN, the deepness comes from layers through time. Furthermore, due to the large hidden space that RNNs can represent (exponential in the number of hidden units), plus the non-linear dynamics that they can model, they can learn to memorize events with longer context, and may be a better fit for speech data.

We describe related work on deep learning and RNNs in speech in Section 2 and our Tandem approach in Section 3. Experimental results and conclusions comprise Sections 4 and 5, respectively.

2. RELATED WORK

The neural-network-based Tandem approach, originally proposed in [6], has been used in many systems to improve recognition performance. In a Tandem system, a base feature such as a mel-scale cepstral coefficients (MFCCs) or perceptual linear prediction (PLP) is used as the input to an MLP trained on relevant labels (typically phones, or monophone HMM states). Then, the outputs of the network can themselves be used as features to a recognizer. Two advantages of the Tandem approach are that non-traditional features can be easily incorporated into a speech recognition system, as shown in [1], and it is robust to noise [7].

2.1. Deep Neural Networks

One of the challenges with the MLP and the DNN is that the objective function is non-convex, and as more hidden layers are added, finding a good local minima becomes more challenging. Motivated

[1]In this paper, we use MLP to denote a single hidden layer Neural Network architecture, whereas DNN implies a deeper architecture with two or more hidden layers. Deep Belief Network (DBN) is used when we use pretraining to train a DNN.

by this problem, DNNs with pretraining based on Restricted Boltzman Machines, denoted as Deep Belief Networks (DBNs), were introduced in [8] and have been applied to several fields such as computer vision (see [9]), phone classification (see [10]), speech recognition [2, 3, 11, 4], and speech coding (see [12]). The new idea is to train each layer independently in a greedy fashion, by sequentially using the hidden variables as observed variables to train each layer of the deep structure. Recently, the use of DNNs with no pretraining has also been studied [4, 11].

2.2. Recurrent Neural Networks

Recurrent Neural Networks were first applied to speech recognition in [5]. RNNs are powerful models that can model non-linear dynamics through connections between hidden layers, as can be seen in Figure 1. One of the key challenges for training RNNs is that long term dependencies are difficult to capture since vanishing gradients over time preclude the update of weights from the far past. Back Propagation Through Time and approximations to it have been used before. More recently, applications in Language Modeling [13] and advances in optimization [14] have seen state of the art performance by the usage of RNNs for sequence modeling.

2.3. Pretraining and optimization

In this paper, we further explore the interaction between pretraining and the optimization method to learn the model parameters for both DNN and RNN models for robust speech recognition. Analysis on why pretraining is useful has been discussed in both the machine learning [15] and speech recognition [4, 11] communities. Given that the training of RNNs is more problematic than of DNNs due to the vanishing gradient problem [16], we developed a new second order optimization algorithm derived from Hessian Free optimization [17].

3. PROPOSED METHOD

Our RNN approach follows the same formulation as in [14]. Unlike in the DNN case, only the current frame without context is used at the observation at time t, $\mathbf{x}_t$. The RNN is able to "remember" context naturally due to its large hidden state representation and its recurrent nature. The architecture is as follows:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1})$$
$$\mathbf{o}_t = \mathrm{softmax}(\mathbf{W}_{oh}\mathbf{h}_t)$$

where the bias terms are omitted for simplicity, $\mathbf{h}_t$ represents the hidden state of the network at time t, and $\mathbf{W}_{hx}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{oh}$ are parameters to be learned. Note that, due to the recursion over time on $\mathbf{h}_t$, the RNNs can be seen as a very deep network with T layers, where T is the number of time steps. We define the initial seed of the network $\mathbf{h}_0$ to be another parameter of our model, and we optimize the cross entropy between the predicted phone posterior output $\mathbf{o}_t$, and the true target (given by forced alignment), similar to how DNNs and MLPs are trained.

As we report in Section 5, there are some considerations in the training of the RNN that are important. First, we pretrain the RNN by "disconnecting" the hidden layers temporally, that is, we first optimize forcing $\mathbf{W}_{hh}$ to be zero (in which case, the training reduces to simple MLP training), and then switch to jointly optimize all parameters. Secondly, the MLPs and DNNs have access to future context of four frames (when using a nine frame context, i.e. $t \pm 4$ frames). We enforce this in the RNNs by delaying the output by four frames. Finally, we constrain the length of the utterances T to be at most 60 (which is equivalent to reset the hidden state $\mathbf{h}$ to $\mathbf{h}_0$ every 60 frames), as this results in more efficient learning in our GPU implementation. However, we have also experimented with not truncating utterances at test time. All these factors are empirically evaluated in Section 5.

The outputs of the MLP, DNN, and RNN provide an estimate of the posterior probability distribution for phones. We apply Karhunen-Loève Transform to the log-probabilities of the merged posteriors to reduce the dimensionality to 32 dimensions and orthogonalize those dimensions. We then mean and variance normalize the features by utterance. Finally, we append the resulting feature vector to the MFCC feature. The augmented feature vector then becomes the observation stream for the decoder, which is described in the next section.

4. EXPERIMENTAL SETUP

For this paper, we use the Aurora2 data set described in [18], a connected digit corpus which contains 8,440 sentences of clean training data and 56,056 sentences of clean and noisy test data. The test set comprises 8 different noises (subway, babble, car, exhibition, restaurant, street, airport, and train-station) at 7 different noise levels (clean, 20dB, 15dB, 10dB, 5dB, 0dB, -5dB), totaling 56 different test scenarios, each containing 1,001 sentences. Since we are interested in the performance of MLP, DNN and RNN features in mismatched conditions, all systems were trained only on the clean training set but tested on the entire test set. In this study, we compare 13-dimensional perceptual linear prediction (PLP) features with first and second derivatives used as input features for either an MLP, DNN, or RNN. This Tandem feature is also appended to a 13-dimensional MFCC with first and second derivatives in all the experiments.

The parameters for the HTK decoder used for this experiment are the same as that for the standard Aurora2 setup described in [18]. The setup uses whole word HMMs with 16 states with a 3-Gaussian mixture with diagonal covariances per state; skips over states are not permitted in this model. This is the setup used in the ETSI standards competition. More details on this setup are available in [18].

The architecture considered for the DNNs and MLPs was inspired by the MNIST hand written digit recognition task [9] and is the same that we used in recent work [2]. Further details on the training procedure can be found in [2], but it is worth noting that, in order for the deep network to use temporal information, a context window totaling nine frames is used. This was found helpful in related work as well [11]. For fairness of comparing DNNs with RNNs, we also include a partial study of how pretraining and differing optimization procedures affect the results (we do so by comparing DNNs with and without pretraining).

5. RESULTS

The details of every system reported in this Section are as follows:

- MLP: As described in [2], we train a 720 hidden unit neural network with stochastic gradient descent (using ICSI's Quicknet), with 9 frames of context using 39 dimensional PLP as input.
- DBN: As described in [2], we train a deep belief network with generative pretraining, and conjugate gradient descent. The

structure of the network is of 500-1000-1500 hidden units, with 9 frames of context using 39 dimensional PLP as input.

- DNN: With the same structure of the DBN, but instead of using pretraining and conjugate gradient descent, we use a modified version of the second order optimizer HF [17].
- RNN: Using an RNN with 1000 hidden units, training segments of 60 frames, with a delayed output of 4 frames, and initialized as a regular MLP (i.e., disconnected). At test time, the network is run on the whole sequence (i.e. the limit of 60 frames is not used).

5.1. Phone Recognition

When training the networks, we leave the first 800 utterances for cross validation (as our training algorithms look at CV error for early stopping). Thus, we get error rates for a phone classification task (without an HMM) from the held out data. In Table 1, we observe how the phone error rate in this frame-by-frame classification of the RNN is comparable to a DNN, and better than the DBN proposed in our previous work. There is, however, a fundamental difference between the DBN and both DNN/RNN: due to the pretraining, we are starting the optimization in a more promising region, which may help finding a more desirable local optima. For small recognition tasks such as Aurora2, this has been shown to generalize better. Thus, under mismatched/noisy conditions, the DBN may still outperform the DNN or RNN.

System	MLP	DBN	DNN[2]	RNN[2]
HU	720	500-1K-1.5K	500-1K-1.5K	1K
PER (%)	15.4%	10.1%	8.3%	8.5%

Table 1. *Phone Error Rate on Cross-Validation Set. The first two columns correspond to the reported PER in [2]. The third column is the same as the second but with no pretraining, and using a better optimization technique. The last column corresponds to an RNN with 1K hidden units.*

5.1.1. Recurrent Neural Network system

In this section, we explore how the RNNs perform when training and testing conditions are changed. As seen in Table 2, adding in the non-linear dynamic term $\mathbf{W}_{hh}$ yields the biggest improvement to the system, showing the usefulness of temporal context. Also, as one can expect, adding future context (which, in the context of DNN/MLP training is done by adding 4 frames of future context to the input), helped. One of the most surprising facts is that long term context seems to help more than we expected: if we restart the RNN state every 60 frames, we achieve 10.2% PER, whereas if we let the RNN run on the whole utterance without restarts, the performance is almost 2% absolute better. This indicates that the RNN is capturing long term effects that yield significantly better PER.

It is worth noting that if we do not use the discriminatively trained disconnected network to initialize the RNN (which is similar to layerwise backpropagation [11]), convergence is rather slow and, even with the second order optimization method, the solution is far from optimal (worse than 40% PER).

[2]Trained using a Hessian Free derived second order method [17].

System	PER
Disconnected RNN ($\mathbf{W}_{hh} = 0$)	18.8%
RNN	11.8%
+ future context (4 frames)	10.2%
+ non truncated testing	8.5%

Table 2. *Phone Error Rate on Cross-Validation Set with different training/testing schemes for RNNs. Error rates are reported for 1000 hidden units.*

5.2. Speech Recognition

Typical results on the Aurora2 test set using the ETSI setup report accuracies (or mean accuracy) across the 8 noises at 7 noise conditions. We do not report accuracies here for two reasons. The first and rather mundane reason is that reporting hundreds of numbers will result in a table too large for the length constraints of this paper. Second, and perhaps more importantly, we do not think that reporting accuracies in general (even with a reduced table) is properly illustrative of the performance of the system. Consider, for instance, a table consisting of results for two systems in two noise conditions, one clean and one extremely noisy. If the baseline achieved a 98% accuracy rate on the clean test and 3% accuracy on the noisy one, and the proposed system achieved a 99% and 1.9% accuracy on the clean and noisy conditions, respectively, one would clearly choose the latter system as that system reduced over half the errors on the clean test while performing roughly similarly on the noisy one (that is, neither really worked in noise). If we simply look at mean accuracy, however, we see that the baseline actually outperforms the compared system. The reduction in errors corresponds fairly well to the common costs of using a system (for instance, how often a system must retreat to a human operator). For this reason, we report WER results, which for many years have been the standard for most speech recognition tasks.

For this paper, we average WER across noises and report scores for each noisy condition. Finally, all results are significant with a p-value of 0.002 using the differences of proportions significance test.

5.2.1. Tandem systems

For this set of experiments, we concatenate our processed posterior probabilities with MFCC features (i.e. Tandem system). As noted in [2], it is generally better to append MFCC features to the discriminatively trained networks (except for very noisy conditions, in which case MFCCs degrades performance of MLP/DNN/RNN).

In Table 3 we find the results for our MFCC baseline, and several Tandem systems. As can be seen, adding a deep network in Tandem with MFCCs clearly outperforms the MFCC baseline. It is interesting that the DBN is generally more competitive under mild noise conditions than DNN and RNN, presumably due to the generative initialization, which is known to yield better generalization (specially in small datasets). The RNN outperforms every model under clean speech, which is interesting and should be further explored with a larger speech database. The fact that the RNN is also better under very noisy conditions may indicate that, in those cases, longer term dependencies that the MLP/DBN/DNN models cannot capture may be necessary. Lastly, the number of hidden units for the RNN did not seem to have a big effect: 200 units seemed to underfit the data, and 1500 units were a bit better under high noise condition, but worse on clean (presumably due to overfitting). We expect this

parameter to be more relevant when the training set exhibits more variance and is larger, but we leave this for future work.

SNR	MFCC	MLP	DBN	DNN	RNN
Clean	1.60%	1.56%	0.88%	0.78%	**0.70%**
20dB	5.33%	3.68%	**2.69%**	3.05%	3.59%
15dB	14.77%	7.47%	**6.35%**	**6.38%**	6.89%
10dB	36.59%	16.63%	15.26%	**14.61%**	**14.77%**
5dB	66.65%	36.82%	34.34%	32.09%	**30.94%**
0dB	86.98%	63.80%	61.43%	58.98%	**57.19%**
-5dB	94.01%	87.09%	85.71%	84.42%	**81.51%**

Table 3. *Average WER for several systems under different noise conditions. The first three columns correspond to the reported results in [2]. Bold numbers indicate best performance. Note that, as before, DNN and RNN use the second order optimization method.*

6. DISCUSSION AND FUTURE WORK

In this paper, we extend our work on deep learning, in which we already studied how the deep models integrate with MFCC using the Tandem approach, and the deep model is robust to different noise conditions present in the Aurora2 dataset.

Our first extension is to study the effect of pretraining of our previously proposed model. We found that the standard approach to deep learning, that is, DBNs trained using pretraining and standard conjugate gradient descent, and the DNNs with no pretraining but trained with a second order optimization method performed similarly under clean condition. DBNs were, however, able to perform better under mild noise conditions, which can be explained by the nature of pretraining to better generalize to unseen conditions during training.

We also revisit RNN, a model that seems natural for sequential data such as speech, but poses a difficult optimization problem. Using our proposed pretraining and the second order method, we were able to successfully train the RNN and use its outputs in a Tandem approach, which yields the best clean condition score for HMM Tandem system in the Aurora2 set, and better performance on the noisier conditions (although the WER in those scenarios is still too high for most practical applications).

Given the performance seen on clean speech, one of the lines of research is to apply this approach to a larger dataset. When orders of magnitude more data is available, the relative differences of each model may change (e.g. pretraining is not as valuable once the amount of training data is large enough).

Since both the RNN and DNN improve performance, one could combine the model to create a deep recurrent neural network, in which multiple hidden layers are used. Also, instead of using Tandem, we will try using a hybrid system, which will replace the GMM emission model by the RNN. Lastly, predicting subphone states of the HMM instead of phone posteriors is beneficial [4, 14] and, with more data available, could yield further improvements.

7. ACKNOWLEDGEMENTS

We would like to thank Prof. Nelson Morgan for useful discussions on using neural networks in the pipeline of speech recognition. We would like to acknowledge the Microsoft Research Fellowship and the National Defense Science and Engineering Graduate Fellowship (NDSEG) for helping to support this project.

8. REFERENCES

[1] H. Hermansky and S. Sharma, "Temporal patterns (TRAPS) in ASR of noisy speech," in *ICASSP*, 1999.

[2] O. Vinyals and S. V. Ravuri, "Comparing multilayer perceptron to Deep Belief Network Tandem features for robust ASR," in *ICASSP*, 2011.

[3] A. Mohamed, D. Yu, and L. Deng, "Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition," in *Interspeech*, 2010.

[4] G. 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*, 2012.

[5] T. Robinson, M. Hochberg, and S. Renals, "IPA: Improved phone modelling with recurrent neural networks," in *ICASSP*. IEEE, 1994.

[6] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem Connectionist Feature Extraction for Conventional HMM Systems," in *ICASSP*, 2000.

[7] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky, "Feature Extraction Using Non-Linear Transformation For Robust Speech Recognition On The Aurora Database," in *ICASSP*, 2000.

[8] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[9] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[10] A. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop*, 2009.

[11] F. Seide, G. Li, and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks," in *Interspeech*, 2011.

[12] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton, "Binary Coding of Speech Spectrograms Using a Deep Auto-encoder," in *Interspeech*, 2010.

[13] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent Neural Network based Language Model," in *Interspeech*, 2010.

[14] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *ICML*, 2011.

[15] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] O. Vinyals and D. Povey, "Krylov Subspace Descent for Deep Learning," in *AISTATS*, 2012.

[18] H.G. Hirsch and D. Pearce, "The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR: Challenges for the Next Millennium*, 2000.