COMPARING MULTILAYER PERCEPTRON TO DEEP BELIEF NETWORK TANDEM FEATURES FOR ROBUST ASR

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

[1]International Computer Science Institute, Berkeley, CA, USA
[2]EECS Department, University of California - Berkeley, Berkeley, CA, USA

vinyals@icsi.berkeley.edu, ravuri@icsi.berkeley.edu

ABSTRACT

In this paper, we extend the work done on integrating multilayer perceptron (MLP) networks with HMM systems via the Tandem approach. In particular, we explore whether the use of Deep Belief Networks (DBN) adds any substantial gain over MLPs on the Aurora2 speech recognition task under mismatched noise conditions. Our findings suggest that DBNs outperform single layer MLPs under the clean condition, but the gains diminish as the noise level is increased. Furthermore, using MFCCs in conjunction with the posteriors from DBNs outperforms merely using single DBNs in low to moderate noise conditions. MFCCs, however, do not help for the high noise settings.

Index Terms: Automatic Speech Recognition, Deep Belief Network, Multilayer Perceptron

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 apply a Tandem system, where phone posterior probabilities are appended to standard MFCCs.

Typically, these posteriors are generated through some discriminative process, and MLP have successfully been used in speech for many years. More recently, new research into the training strategies for deep networks have allowed for improved performance in many machine learning and pattern recognition tasks, as described in Section 2. A key insight into train such deep networks, which have more modeling power than the networks with flatter structure, is that standard techniques such back propagation frequently lead to suboptimal parameter settings. Recently, a new learning strategy has been proposed, based on a greedy, generative pre-training process using Restricted Boltzman Machines (RBMs), where each layer of the network is trained generatively through contrastive divergence[1]. This provides better initial parameters that can then be used for fine tuning all the parameters of the network jointly in a discriminative fashion.

Since Deep Belief Networks share the same output structure as multilayer perceptrons, we endeavored to show how DBNs can be employed in a Tandem framework. Furthermore, we compare the newly proposed DBNs to standard MLPs, and we analyze how the deep structure, which has been proven to help in phone recognition [2], is affected under several noise conditions present in the Aurora corpus.

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

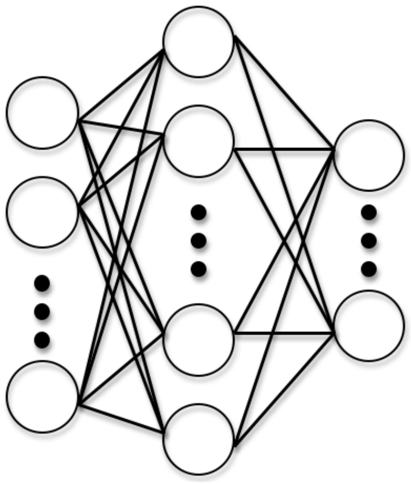
Fig. 1. *Structure of a Multilayer Perceptron.*

2. RELATED WORK

The neural-network-based Tandem approach, originally proposed in [3], has been used in many systems to improve recognition performance. In a Tandem system, a base feature such as a mel-scale cepstral coefficients or perceptual linear prediction (PLP) is discriminately trained by a multilayer perceptron. Then, the outputs of the features can themselves be used as features to a recognizer. Two advantages of the Tandem approach are that non-traditional features can be used, as shown in [4] and it is robust to noise, as shown in [5],

2.1. Deep Belief Networks

The problem with MLPs is that the objective function is non-convex, and as more hidden layers are added, finding a good local minima is more challenging. Motivated by this problem, DBNs were introduced in [1] and have been applied in several fields such as computer vision (see [6]), phone classification (see [7, 2]), and speech coding (see [8]). 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.

2.1.1. Restricted Boltzman Machines

It is out of the scope of this paper to discuss all the details for the training of DBNs or RBMs. An extensive technical report can be

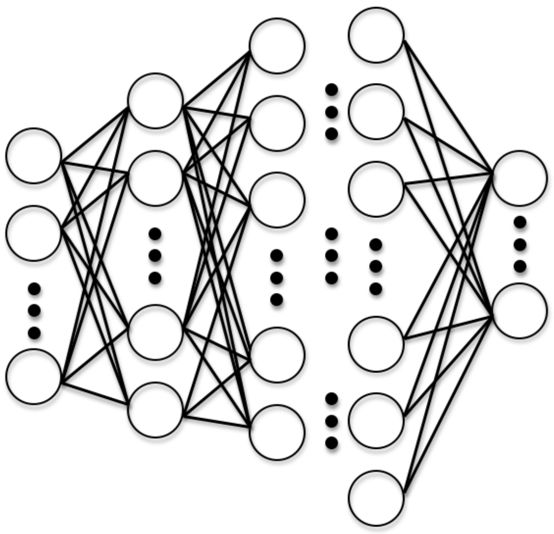

Fig. 2. *Structure of a Deep Belief Network.*

found in [9]. In this particular approach, note that the Restricted Boltzman Machine (RBM), a type of Markov random field, has one layer of stochastic visible units and one layer of stochastic hidden units. The graph is bipartite, since no connections are allowed between visible units (or hidden units), but each visible unit is connected to all hidden units. Since our input comes from PLP features (normalized to zero mean and unit variance), we model the first visible layer of the DBN as a Gaussian random variable. The rest of the hidden layers are considered as Bernoulli random variables.

Training for a single RBM in a generative, maximum likelihood way, requires to compute the following gradient:

$$\Delta w_{i,j} = < v_i h_j >_{data} - < v_i h_j >_{model}$$

where v_i and h_j are random variables corresponding to the visible and hidden layer respectively. Note that the computation of the expectation under the data is trivial, but the expectation under the model would require to run a Gibbs sampler to get the gradients for every single update. Since this would not be practical, [1] proposed to use contrastive divergence, which approximates the expectation by running a single step of the Gibbs sampler. In general, one could run n steps of the Gibbs sampler to obtain a better approximation for the gradient, but in our setting a single step seems sufficient.

2.1.2. *Deep Belief Networks*

Once the pre-training step is performed on all the levels of the network by treating each pair of variables as a RBM and using the hidden variables in one step as the visible variables for the next step in a greedy fashion, we use the weights to initialize the fully connected DBN. In this step, we add an additional layer corresponding to the class labels activations. During this stage, we fine tune all the weights jointly using back-propagation, by maximizing the frame level cross entropy between the predicted and true probability of the class labels in a discriminative fashion. Note that this last step is very similar to how MLPs are trained.

3. EXPERIMENTAL SETUP

For this paper, we use the Aurora2 data set described in [10], 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 and DBN 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 discriminatively trained either by an MLP or a DBN. This "Tandem" feature is also appended to a 13-dimensional MFCC with first and second derivatives in some experiments.

The architecture considered for the Deep Belief Network was inspired by the MNIST hand written digit recognition [6]. In this case, however, we have 351 input units (corresponding to PLP features with a context window of nine frames), and three hidden layers with 512, 1024, and 1536 hidden units. The output layer correspond to phone activations and has 56 units. Note that the total number of parameters is about 2.4M. Due to hardware limitations, we only train on 500K frames, which is a fraction of the Aurora training set. The frames were randomized, and a set of 800 utterances were held out as cross validation (CV) (which was also used in MLP training). We performed 40 epochs for each of the pre-training stages, and used early stopping during fine tuning to avoid overfitting. We noted that, even though the number of parameters is higher than the number of training examples, CV error seems to be stable and overfitting did not seem to be an issue.

We compare Deep Belief Network features to the canonical multilayer perceptron features. The architecture of the MLP is 351 input units, which correspond to the same PLP features with 9 frames of context used as the input of the DBN, a varying number of hidden units, and 56 output units. We experimented with different sizes of the hidden layer, but found in that 720 hidden units yields the best performance for recognition. We use the standard back-propagation algorithm with weights initialized randomly and employ the "newbob" learning rate schedule, which learns at a fixed rate until the accuracy on a cross-validation set fails to improve by more than 0.5% between two epochs. Then, the MLP continues to halve the learning rate at each until the CV accuracy again fails to improve by more than 0.5%, at which point the algorithm terminates.

The outputs of the MLP and DBN provide an estimate of the posterior probability distribution for phones. We apply Karhunen-Loève Transform to the log-probabilities of the merged MLPs 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 HTK decoder.

The parameters for the HTK decoder are the same as that for the standard Aurora2 setup described in [10]. 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, from which AFE was created. More details on this setup are available in [10].

4. RESULTS

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 HMM) from the held out data by using our systems. In Table 1 we observe how the phone error rate in this frame-by-frame classification is already significantly lower for the DBNs. Thus, we expect gains by using the new DBN Tandem system with respect to the MLP Tandem system.

4.1. Phone Recognition

System	MLP 320HU	MLP 512HU	MLP 720HU	MLP 1100HU	DBN
PER (%)	15.4%	15.4%	15.4%	15.4%	**10.1%**

Table 1. *Phone Error Rate on Cross-Validation Set.*

4.2. 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.[1] 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.

4.2.1. Stand-alone systems

In the first set of recognition experiments, we substitute MFCC with posteriors coming from MLP or DBN systems (i.e., we do not use Tandem) trained on PLP features. The motivation for this experiment is two-fold. The first is to compare the performance of MLPs and DBNs without any ancillary information. The other is to verify that indeed MFCC still add information that makes recognition better.

[1]The full table, however, is available at http://www.icsi.berkeley.edu/~ravuri/DBNMLPresults.html

We tested MLP with varying hidden units from 320 to 1100, although increasing the number of hidden units (and, thus, parameters) did not necessarily improve the WER. This is a problem that DBNs seem to circumvent, perhaps because of the pre-training through RBMs. Table 2 shows the overall WER for the best performing MLP system versus the DBN, and Table 3 shows relative improvements. As we can see, error rate for DBN system is significantly lower than for MLPs in the low noise setting, while the performance of the two systems is comparable in the high noise one. Also note that the MLP system performs worse than the MFCC baseline while its DBN counterpart outperforms baseline system. This last observation is compatible with recent work [2].

SNR	MFCC	MLP 720 HU	DBN
clean	1.60%	2.47%	1.26%
20dB	5.33%	6.30%	3.90%
15dB	14.77%	10.32%	7.82%
10dB	36.59%	17.97%	15.00%
5dB	66.65%	31.34%	28.33%
0dB	86.98%	51.62%	50.01%
-5dB	94.01%	74.84%	74.62%

Table 2. *Average WER for different Systems.*

SNR	MLP 320HU	MLP 512HU	MLP 720HU	MLP 1100HU	DBN
clean	-78.20%	-94.69%	-54.14%	-68.59%	21.48%
20dB	-31.05%	-30.30%	-18.15%	-20.83%	26.78%
15dB	25.28%	27.84%	30.11%	28.17%	47.08%
10dB	48.61%	50.25%	50.89%	49.04%	59.01%
5dB	52.82%	53.88%	52.98%	51.64%	57.49%
0dB	40.99%	41.70%	40.63%	38.94%	42.48%
-5dB	20.65%	21.30%	20.40%	18.52%	20.63%

Table 3. *Percentage Improvement Relative to MFCC Baseline.*

4.2.2. Tandem systems

For this set of experiments in which we concatenate our posterior probabilities with MFCC features (i.e. Tandem system), the overall error rates are reduced with respect to the previous systems on low noise scenario. Tables 4 and 5 are analogous to the ones in Section 4.1.1.

Note that, generally, the use of MFCC features helped in the low noise conditions (Clean, 20dB, 15dB), while in noisy cases the MLP and DBN system perform better without MFCC (seen highlighted in bold in Table 6). This is not surprising given that there is a large spectral mismatch between training and testing on noisy conditions, whereas the networks are learning "higher" level spectral representations, and thus, are more robust to noise.

5. DISCUSSION AND FUTURE WORK

The performance of the DBN, used as Tandem features, is quite promising, especially in the view that the DBN was trained on only 500K frames of data, while the MLP was trained on the full training set, which consisted of 1.4M frames. In the phone recognition and two ASR tasks, the features generated from the DBN were overall

SNR	MFCC	MLP 720HU	DBN
Clean	1.60%	1.56%	0.88%
20dB	5.33%	3.68%	2.69%
15dB	14.77%	7.47%	6.35%
10dB	36.59%	16.63%	15.26%
5dB	66.65%	36.82%	34.34%
0dB	86.98%	63.80%	61.43%
-5dB	94.01%	87.09%	85.71%

Table 4. *Comparison of Average WER for MFCC, MFCC+MLP, and MFCC+DBN Tandem systems.*

SNR	MLP 320HU	MLP 512HU	MLP 720HU	MLP 1100HU	DBN
Clean	-14.03%	-0.08%	2.12%	-11.91%	44.59%
20dB	20.96%	23.23%	30.84%	16.29%	49.43%
15dB	46.35%	47.70%	49.45%	41.62%	57.01%
10dB	53.47%	53.22%	54.55%	46.81%	58.28%
5dB	46.77%	47.01%	44.75%	37.26%	48.48%
0dB	33.14%	34.25%	28.30%	23.21%	30.96%
-5dB	13.14%	16.28%	7.36%	8.31%	8.83%

Table 5. *Percentage Improvement Relative to MFCC Baseline for Tandem systems.*

SNR	MLP	DBN	MLP+MFCC	DBN+MFCC
Clean	2.47%	1.26%	**1.56**%	**0.88**%
20dB	6.30%	3.90%	**3.68**%	**2.69**%
15dB	10.32%	7.82%	**7.47**%	**6.35**%
10dB	17.97%	**15.00**%	**16.63**%	15.26%
5dB	**31.34**%	**28.33**%	36.82%	34.34%
0dB	**51.62**%	**50.01**%	63.80%	61.43%
-5dB	**74.84**%	**74.62**%	87.09%	85.71%

Table 6. *Performance comparison of non-Tandem versus Tandem systems (MLP refers to MLP 720HU in previous Tables).*

the best-performing. This suggests that DBNs could be used as a replacement for MLPs, especially in situations when the training set is small.

There are, however, some caveats to note to the study. First, the number of parameters to the DBN is 2.4M, while the biggest MLP - the one with 1100 units in the hidden layer - only has 447K parameters. The MLP with this many parameters performed more poorly on the speech recognition task compared to its counterparts with smaller hidden layers, suggesting some overfitting with the MLP.

Moreover, we are not sure what part of the DBN allows it to outperform the MLP. The pre-training step used in the DBNs to prevent overfitting may be one part, but in initial experiments training an MLP using weights initialized by the pre-training stage of the DBN yielded no change in performance compared to the MLP without pre-training. This suggests that pre-training may only help when the structure is deep, and that the deep structure of the DBNs is what makes for a better overall performance.

Furthermore, it would be interesting to see how the MLP compares to the DBN when there exists a large amount of training data. It may be that the structure and training of the DBN allows it to not overtrain with small amount of data and large number of parameters, but this property may not matter with a large amount of training data.

Finally, it is interesting to note that on very noisy conditions – at 5dB, 0dB and -5dB SNR – MFCC features did not help, and non-Tandem systems where the network features were used alone proved to be more robust (e.g. at -5dB, MFCC has 94% error, DBN with Tandem is at 86% while DBN alone does better at 75% error).

6. ACKNOWLEDGEMENTS

We would like to thank Prof. Nelson Morgan and George Dahl for helpful comments on this paper. Moreover, we would like to acknowledge the National Science Foundation (NSF) and the National Defense Science and Engineering Graduate Fellowship (NDSEG) for helping to support this project.

7. REFERENCES

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

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

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

[4] Hynek Hermansky and Sangita Sharma, “Temporal patterns (traps) in asr of noisy speech,” in *in Proc. ICASSP*, 1999, pp. 289–292.

[5] Sangita Sharma, Dan Ellis, Sachin Kajarekar, Pratibha Jain, and Hynek Hermansky, “Feature extraction using non-linear transformation for robust speech recognition on the aurora database,” in *Proc. ICASSP*, 2000, pp. 1117–1120.

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

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

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

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

[10] 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.