Autoencoder based multi-stream combination for noise robust speech recognition

Sri Harish Mallidi[1], *Tetsuji Ogawa*[3], *Karel Vesely*[4], *Phani S Nidadavolu*[1], *Hynek Hermansky*[1,2]

[1]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, U.S.A
[2]Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, .U.S A
[3] Waseda University, Department of Computer Science and Engineering, Tokyo, Japan
[4] Brno University of Technology, Speech@FIT group, Brno, Czech Republic

Abstract

Performances of automatic speech recognition (ASR) systems degrade rapidly when there is a mismatch between train and test acoustic conditions. Performance can be improved using a multi-stream framework, which involves combining posterior probabilities from several classifiers (often deep neural networks (DNNs)) trained on different features/streams. Knowledge about the confidence of each of these classifiers on a noisy test utterance can help in devising better techniques for posterior combination than simple sum and product rules [1]. In this work, we propose to use autoencoders which are multilayer feed forward neural networks, for estimating this confidence measure. During the training phase, for each stream, an autocoder is trained on TANDEM features extracted from the corresponding DNN. On employing the autoencoder during the testing phase, we show that the reconstruction error of the autoencoder is correlated to the robustness of the corresponding stream. These error estimates are then used as confidence measures to combine the posterior probabilities generated from each of the streams. Experiments on Aurora4 and BABEL databases indicate significant improvements, especially in the scenario of mismatch between train and test acoustic conditions.

Index Terms: speech recognition, human-computer interaction, computational paralinguistics

1. Introduction

State-of-the-art speech recognition technology performs reasonably well when test condition is similar (matched) to the training condition. Performance of the automatic speech recognition (ASR) system degrades rapidly when there exists a mismatch between test and train conditions. Robustness of ASR systems can be increased using multi-stream ASR framework [2, 3]. Multi-stream ASR framework involves training several classifiers independently on different signal representations (acoustic features) and combining the decisions from the classifiers during testing. State-of-the-art classifiers used in ASR tasks are DNNs. The central issue in multi-stream ASR framework is combination of decisions (e.g. posterior probabilities, N-best hypothesis, recognition outputs etc) from multiple streams. In this work, we focus on frame-based combination of posterior probabilities of DNNs from multiple streams. A good combination rule should assign more weight to decisions of DNNs which are robust to given acoustic condition. This is achieved by using weights proportional to **estimated** performance of DNN in the corresponding stream [8, 9]. The weights are estimated based on the premise that a DNN is at its best when applied to its training data. Deviations of stochastic regularities derived from training data degrade its performance. A general framework for performance estimation of DNN based classifiers is as follows:

- Model the activations of DNN on the train data
- Performance on test data is estimated by measuring the deviation of test activations from the model.

Okawa et. al., Misra et. al. [8, 9] observed that as the noise in test data increases, the output posterior probability distribution from the multi-layer perceptron (MLP), trained on clean data, converges to non-informative, uniform distribution. This results in increase in the entropy of posterior distribution. Based on this observation, inverse of entropy was proposed as a measure of performance. Clearly, inverse entropy combination rule is a special case of the general framework. In inverse entropy rule, the assumption is output softmax layer activations on training

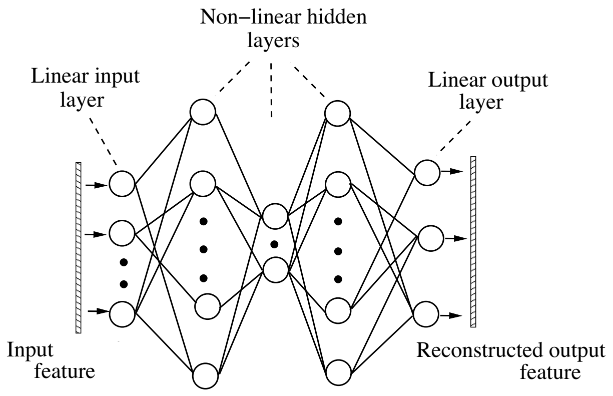

Figure 1: A five layered autoencoder, with 3 non-linear hidden and 2 linear visible layers. Architecture of autoencoder used in this paper is $\{Y \times 512 \times 24 \times 512 \times Y\}$, where Y corresponds to input feature dimension.

This work was supported in parts by the National Science Foundation via award number IIA-0530118, .Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013, by Google via Google faculty award to Hynek Hermansky. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Google, NSF, IARPA, DoD/ARL, or the U.S. Government.

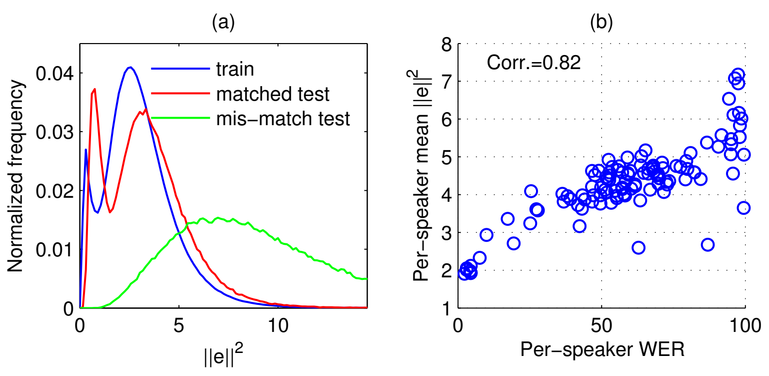

Figure 2: (a) Illustration of property of autoencoder useful to distinguish matched data and mis-matched data. (b) Per-speaker mean $||e||^2$ *vs* WER.

data, result in a peaky, low entropy posterior distributions. So performance on a test sentence is estimated to be proportional to inverse of entropy values.

In this work, we propose to use an autoencoder to model the activations of DNN. Autoencoders are feed-forward neural networks, used for modeling complex data distributions [4, 5, 6]. We train the autoencoders to reconstruct DNN activations of training data. The reconstruction error on test data is used as measure of DNN confidence on the test data. Autoencoder based method can be thought as a generalization of entropy method, since there are no assumptions on the structure of DNN outputs of train data. Also, autoencoders can be used to model, activations at the hidden layer as well, which is not possible in rule based methods like entropy.

The reminder of the paper is organized as follows: Section 2 provides details about architecture, training criterion and basic properties of autoencoder. Section 3 introduces multi-stream ASR system, conventional combination rules and proposed combination rule based on autoencoder. Section 4 introduces the experiments performed using autoencoder based combination rule on Aurora4 and Babel databases, and Section 5 concludes the paper.

2. Autoencoder

Autoencoder is a multi-layered feed-forward neural network, used in the context of unsupervised learning. During the training process, parameters of the network are optimized to minimize squared error cost between a target vector and output vector from the autoencoder. The targets used to train the network are inputs themselves. The cost function used to optimize the network parameters ($\mathcal{W}$) is shown in the following equation:

$$\min_{\mathcal{W}} \mathbf{E}||\mathbf{x} - \hat{\mathbf{x}}||_2^2 \qquad (1)$$

where $\mathbf{x}$ is input vector and $\hat{\mathbf{x}}$ is output vector from the network. Figure 1 shows the architecture of autoencoder used in the present work. An autoencoder with more than one non-linear hidden layer is shown to capture complex, non-linear manifolds present in the training data [5, 14]. In order to avoid a trivial identity mapping (weights of network equal to unit matrix), the number of nodes in the second hidden layer are chosen to be fewer than input (or output) layer.

Since the network is trained to minimize the reconstruction error, a vector sampled from distribution of the training data will have a low reconstruction error compared to vector drawn from a different distribution. This property is illustrated in figure 2, which shows distribution of l_2 norm of reconstruction error vectors ($||e||^2$), computed from training data, data similar to training data, and data that deviates from the training data. Figure 2 (a) illustrates that reconstruction error is a good indicator of the mis-match between training data and test data. Further to evaluate the ability of reconstruction error to predict the performance, we computed average reconstruction error for each speaker in Aurora4 test set and correlated with average word error rate of that speaker. A total of 112 data points, computed over 14 test conditions of Aurora4 database, was used in the plot. Figure 2 (b) shows the correlation between per-speaker reconstruction error *vs* per-speaker word error rate. It is evident from the plot reconstruction error correlate well with WER, and can be used to estimate the performance. Details about the training set, features, etc used to generate the plot are provided in 4.1.

3. Multi-stream ASR

Multi-stream framework involves extracting multiple features from the acoustic signal. Similar to previous studies [10, 11, 12, 13], we used short-term spectral and long-term temporal modulation features to study the effectiveness of proposed combination rule. We used perceptual linear prediction (PLP [17]) features to capture short-term spectral information and MRASTA [18] features are used to capture log-term temporal information. In each critical band, one second long, energy envelope is filtered using filters, representing first derivative $G1 = [g1_{\sigma_i}]$ and second derivative $G2 = [g2_{\sigma_i}]$ of Gaussian function with variance $\sigma_i \in \{0.8, 1.2, 1.8, 2.7, 4, 6, 8.5, 13\}$. These variance values result in a filter-bank which is equally spaced in logarithmic modulation frequency domain.

During training phase of the system, two separate DNNs are trained using MRASTA and PLP features. During test phase, MRASTA and PLP features are extracted from the given acoustic signal, and forward passed through their respective DNNs. Similar to previous studies [9, 10, 12, 13], the posterior probability vectors are combined using at each time frame. Lattice generation and decoding is then performed on the combined posterior probability vector. Figure 3 depicts the architecture of multi-ASR system.

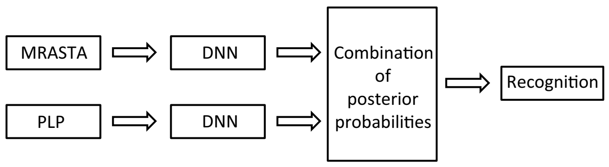

Figure 3: Multi-stream ASR architecture. Two separate DNNs are trained using PLP and MRASTA features. State posterior probabilities obtained from the 2 DNNs are fused, and then used for recognition.

3.1. Baseline combination rules

Various combination rules have been proposed in literature to combine posterior probabilities obtained from neural networks: sum, product, maximum, minimum rules [1]. In this section we describe a few of the rules. Let us consider X_{at} and X_{bt} as the two feature streams at a time t. Let $p_{at} \equiv \mathrm{P}(s|X_{at})$ and $p_{at} \equiv \mathrm{P}(s|X_{bt})$ denote the posterior probability of HMM states obtained from DNNs trained on the two feature streams, X_{at} and X_{bt}, respectively. Sum and product rules [1] combine the posterior probabilities as follows:

$$\begin{aligned} p_{ct} &= (p_{at} + p_{bt})/Z_t \\ p_{ct} &= (p_{at} \times p_{bt})/Z_t \end{aligned}$$

where Z_t is a normalizing constant, used to make p_{ct} as a valid distribution. Motivation for sum and product rule is given in [1]. Sum rule is a simple combination rule, based on the assumption that all classifiers are equally confident about their decisions. Product rule can be considered a log-linear average of posterior probabilities.

In [9], Entropy of MLP output is used as confidence measure related to how feature streams are affected by noise or mismatch. It was observed that value of entropy, $H(p_{at}) = -\sum_i p_{iat} \log\, p_{iat}$, decreases with SNR of test data. This indicates that posterior converges to a non-informative, uniform distribution over the set of speech classes. Inspired by this findings, [9] proposed a linear weighting scheme, referred to as *inverse entropy* combination. The *inverse entropy* combination is given as,

$$p_{ct} = (w_{at}\, p_{at} + w_{bt}\, p_{bt})$$

where the weights are inversely proportional to value of entropy, i.e.

$$w_{at} = \frac{1/H(p_{at})}{1/H(p_{at}) + 1/H(p_{bt})}, w_{bt} = \frac{1/H(p_{bt})}{1/H(p_{at}) + 1/H(p_{bt})}$$

The sum rule is a special case of inverse entropy combination, when both the streams have equal entropy.

3.2. Proposed combination rule:

In this section, we describe procedure to map reconstruction error obtained from autoencoder to weight associated with each stream. For each stream, we train an autoencoder on the DNN activations of the training data. During testing phase, for each stream, we compute the reconstruction error of activations using the stream's autoencoder. Since, high reconstruction indicates large mis-match between test and train datas, we choose the following weight assignment:

$$w_{at} = \frac{1/||e_{at}||^2}{1/||e_{at}||^2 + 1/||e_{bt}||^2}, w_{bt} = \frac{1/||e_{bt}||^2}{1/||e_{at}||^2 + 1/||e_{bt}||^2}$$

	Correlation with WER
Log. phone posteriors	0.61
TANDEM	0.80
TANDEM_LDA	**0.82**

Table 1: Comparison various input representations for training Autoencoder.

where $||e_{at}||$ and $||e_{bt}||$ are l_2 norms of reconstruction error vectors corresponding to streams a and b, at time t.

4. Experiments

4.1. Noisy speech recognition experiments

The Aurora4 task is a small scale (14 hour), medium vocabulary speech recognition task, aimed at improving noise and channel robustness [19]. Aurora4 database is based on the DARPA Wall Street Journal (WSJ0) corpus which consist of clean recordings of read speech, with 5000 word vocabulary size. The training set consists of 14 hours of clean speech, from 83 speakers, sampled at 16000 Hz. The test set contains simultaneous recordings in 14 different acoustic conditions. These test sets are usually grouped into 4 subsets: clean (1 test case, group A), additive noise (6 test cases, group B), clean with channel distortion (1 test case, group C) and additive noise with channel distortion (6 test cases, group D). The six additive noise conditions are "street", "babble", "train", "car", "restaurant", "airport", with varying signal to noise ratio levels from 5 to 15 dB. Each test condition contains 330 recordings with a total of 40 minutes of speech.

We used hidden Markov model-deep neural network (HMM-DNN) system based ASR system is used for the experiments. The system is implemented using Kaldi speech recognition toolkit [20]. We used 6 hidden layer DNNs. The DNNs are pre-trained using RBM algorithm [21] and fine-tuned using cross-entropy cost function. The targets used for fine-tuning are on context dependent tri-phone states, generated using a HMM-GMM system trained on MFCC features.

4.1.1. Optimal features for Autoencoder training

In order to identify best feature to the train autoencoder, we experimented with 3 feature representations extracted at the output of DNN. These are:

- **Log. phone posterior features:** Phone posterior probabilities are obtained by merging context dependent state posteriors corresponding to the phoneme. Log. phone posteriors are estimated by transforming phone posteriors using logarithm.
- **TANDEM:** Pre-softmax outputs of DNN, trained on context dependent HMM state targets, are transformed using principal component analysis (PCA). The PCA transformation matrix is estimated on the same data used to train the DNN.
- **TANDEM_LDA:** In this feature representation, we transform the pre-softmax outputs using multi-class linear discriminant analysis. The context dependent states are used as classes for estimation of LDA transformation.

Table 4.1 shows the correlation of reconstruction error, computed from autoencoders trained on each of these features. Correlation is computed between mean reconstruction error of a speaker and mean WER of that speaker. It is evident from the

	A	B	C	D
MRASTA	3.70	41.61	17.21	59.03
PLP	3.40	55.39	42.50	71.79
Sum	3.05	41.05	18.92	59.17
Product	**3.03**	39.04	25.33	60.63
Inverse Entropy	3.03	37.54	18.92	56.82
AE_TANDEM_LDA	3.19	**35.95**	**15.17**	**54.42**

Table 2: Comparison of WER (%) of proposed combination technique with baseline combination techniques, on Aurora4. AE_TANDEM_LDA refers to reconstruction error computed from autoencoders trained on TANDEM_LDA features. Test set A corresponds to clean condition, B, C and D corresponds to test set with additive, convolutive and additive+convolutive distortions, respectively.

table that, autoencoder can be used to predict the WER, regardless of feature representation used for its training. The reason for the observed trend in the table might be due to amount of HMM state level discriminative information retained by each transformation. Log. phone posterior has the lowest state level discriminative information of all the 3 features. TANDEM features retain some of the information lost by Log. phone posteriors. In TANDEM_LDA, the discriminative information present in pre-softmax outputs is further emphasized by the additional LDA transformation.

4.1.2. Comparison with baseline combination rules

Table 2 show the results of PLP and MRASTA systems in the 4 test sets of Aurora4 database. In clean condition, system trained on PLP features performs better than the one trained on MRASTA features. Whereas, the performance of system trained on MRASTA features is significantly better in mis-matched conditions, especially in conditions which have convolutive distortions due to microphone variations (test C and D).

Out of the combination rules, product and Inverse entropy combination rules achieve best performance in matched condition (test set A). In mis-matched conditions, sum and product rules doesn't always guarantee WER values lower than individual streams. In test sets C and D, sum and product rule have WERs higher than MRASTA stream. This might be due assumption that all classifiers degrade to the same extent, irrespective of acoustic condition in test data. Inverse entropy combination is improving the performance over both the streams in all the test sets, except test set C. This might be due to significant difference between performance of MRASTA and PLP features in test set C, where MRASTA features has 60 % relative improvement over PLP features.

In matched and mis-matched conditions, results obtained by combination using autoencoder are better than results obtained by each stream independently. This shows that auteencoder based combination never produces results inferior to those of the best individual streams. Of all the combination rules, autoencoder based combination is giving best results in mis-matched conditions. Also, we obtain a 4.2 % relative improvement over inverse entropy in test set B and D, and a significant 19.8 % improvement in test set C. We hypothesize the reason behind this behavior as follows: In the acoustic conditions where is a significant difference between performance of individual streams, proposed combination rule has emphasizes the best stream more aggressively, by assigning more weight.

	MRASTA	PLP	AE_TANDEM_LDA
Home_Office_Mob.	51.6	50.2	49.6
Car_Kit	44.7	44.4	43.5
Public_Place	47.8	46.4	45.9
Street	45.5	43.8	44.2
Vehicle	48.4	48.4	46.9
Microphone	70.0	70.9	68.7

Table 3: WER (%) in various acoustic conditions of Babel Tok Pisin language.

4.2. Speech recognition in real far-field conditions

In this section, we use autoencoders to improve robustness to acoustic mis-matches IARPA Babel data. Babel dataset was chosen as it contains various real world acoustic conditions. These are named as "Vehicle", "Home_Office_Mobile', "Microphone", "Public_Place", "Street" and "Car_Kit". Experiments aimed at improving robustness are performed on Tok Pisin language pack.

We use the following LVCSR pipeline to train individual streams: We first built a GMM system on VLLP data (3 hours) using PLP features. The GMM system is then used to align the FullLP data. The new alignments are used to train two bottleneck MLPs with MRASTA and PLP features. In each stream, a fMLLR-GMM [15] system is trained on bottleneck features corresponding to the stream. The fMLLR features are then used to train a 6 hidden layer DNN, with RBM pre-taining [21]. The training recipe is similar to the one proposed in [16]. The differences between the recipe in [16] and recipe used for here are, the DNNs are not sequence trained and input features are bottleneck features, rather than standard acoustic features. The entire pipeline is referred to as stream. The posterior probabilities obtained from the two streams are combined using autoencoders, as describe in previous section.

Table 3 shows the results in various acoustic conditions in Babel data. Similar to Aurora4 dataset, autoencoder based combination rule performs better than the best individual stream. This shows the generality of the proposed combination rule.

5. Conclusions

In this paper, we have proposed techniques to estimate performance of DNN based classifiers. The technique is based on modeling the distribution of DNN outputs, using autoencoders. Reconstruction error of an autoencoder trained on outputs of DNNs, was shown to correlate well with word error rate. Inverse of reconstruction error values is used as confidence measure of the corresponding DNNs. The confidence measure is applied in multi-stream speech recognition system, using frame level linear weighting of posterior probability vectors from multiple DNNs. The proposed rule is shown to be performing better than previously proposed frame based combination techniques like sum, product and inverse entropy rule, in Aurora4 database. Significant improvements (20 % relative improvement) over baseline techniques are observed in conditions when one of stream is performing significantly better than other stream. Improvement was also observed when tested on more real acoustic mismatch conditions like Babel Tok Pisin database.

6. References

[1] J. Kittler and M. Hatef, “On combining classifiers,” IEEE Trans. on Pattern Analysis and Machine Intelligence, 1998, vol. 20, pp. 226-239,

[2] P. Duchnowski, “A new structure for automatic speech recognition, Ph.D. dissertation, Dept. Electr. Comput. Eng., Massachusetts Inst. Technol., Cambridge, MA, 1992.

[3] H. Bourlard, S. Dupont, H. Hermansky, and N. Morgan, “Towards subband-based speech recognition,” in Proc. EUSIPCO, 1996, pp. 15791582.

[4] M. A. Kramer, (1991). “Nonlinear principal component analysis using autoassociative neural networks.” AIChE Journal, 37(2), 233243.

[5] B. Yegnarayana and S. Kishore, “AANN: an alternative to GMM for pattern recognition”, Neural Networks, pp.459469, 2002.

[6] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, “Auto-Encoder Bottleneck Features Using Deep Belief Networks,” in Proc. ICASSP, March 2012.

[7] S. Tibrewala, and H. Hermansky. “Sub-band based recognition of noisy speech.” Acoustics, Speech, and Signal Processing, IEEE International Conference on. Vol. 2. IEEE Computer Society, 1997.

[8] S. Okawa, E. Bocchieri, and A. Potamianos. “Multi-band speech recognition in noisy environments.” Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on. Vol. 2. IEEE, 1998.

[9] H. Misra, H. Bourlard, and V. Tyagi. “New entropy based multistream combination.” In Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP’03). IEEE International Conference on, vol.1, pp. I-193. IEEE, 2004.

[10] F. Valente, Multi-stream speech recognition based on Dempster-Shafer combination rule, Speech Commun., vol. 52, no. 3, pp. 213222, 2010.

[11] F. Valente and H. Hermansky, Combination of Acoustic Classifiers based on Dempster-Shafer Theory of Evidence, in IEEE ICASSP, 2007, pp. 1129-1132.

[12] Q. Zhu, B. Chen, N. Morgan, and A. Stolcke, “On using mlp features in LVCSR”. In: Proc. ICSLP., 2004

[13] Ch. Plahl, et. al. “Recent improvements of the RWTH GALE mandarin LVCSR system.” In: Proc. Interspeech, 2006, Brisbane, Australia.

[14] C. M. Bishop “Pattern Recognition and Machine Learning”, Springer (2006)

[15] Y. Li, H. Erdogan, Y. Gao, and E. Marcheret, “Incremental online feature space MLLR adaptation for telephony speech recognition,” in ICSLP, 2002.

[16] K. Vesely, A. Ghoshal, L. Burget and D. Povey, “Sequence-discriminative training of deep neural networks”, Interspeech 2013

[17] H. Hermansky, “Perceptual linear predictive (plp) analysis for speech,” The Journal of The Acoustical Sociew of America, 87: 1738-1 752, April 1990.

[18] H. Hermansky, and P. Fousek, “Multi-resolution rasta filtering for TANDEM-based ASR.” In: Proc. Interspeech 2005.

[19] N. Parihar and J. Picone, “Aurora working group: DSR front end LVCSR Evaluation, Technical Report, 2002.

[20] D. Povey, A. et. al., “The Kaldi speech recognition toolkit, in Proc. IEEE ASRU, December 2011.

[21] G. E. Hinton, and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.