INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

A framework for Practical Multistream ASR

Sri Harish Mallidi[1], *Hynek Hermansky*[1,2]

[1]Center for Language and Speech Processing,
[2]Human Language Technology Center of Excellence,
The Johns Hopkins University, Baltimore, U.S.A.

Abstract

Robustness of automatic speech recognition (ASR) to acoustic mismatches can be improved by using multistream architecture. Past multistream approaches involve training large number of neural networks, one for each possible stream combination. During testing phase, each utterance is forward passed through all the neural networks to estimate best stream combination. In this work, we propose a new framework to reduce the complexity of multistream architecture. We show that multiple neural networks, used in the past approaches, can be replaced by a single neural network. This results in a significant decrease in the number of parameters used in the system. The test time complexity is also reduced by organizing the stream combinations in a tree structure, where each node in the tree represent a stream combination. Instead of traversing through all the nodes, we traverse through paths which resulted in a increase in the performance monitor score. Compared to state-of-the-art baseline system, the proposed approach resulted in 13.5 % relative improvement word-error-rate (WER) in Aurora4 speech recognition task. We also obtained an average of 0.7 % absolute decrease in WER in 5 IARPRA-BABEL Year 4 languages.

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

1. Introduction

Performance of state-of-the-art ASR systems can degrade rapidly when there is a mismatch between training data acoustic conditions and test data acoustic conditions. Multistream speech recognition provides an intuitive way to improve robustness of ASR systems to acoustic mis-matches. The fundamental motivation behind multistream recognition is, noise or environmental distortion effects only few parts of the signal space (e.g. frequency bands or spectro–temporal frequency bands). The portions which are less corrupted can be identified and decisions from these streams can be emphasized.

In order to build a robust multistream system, we need streams which are localized in signal space as much as possible [1]. This results in a large number of streams. Full Combination MultiStream (FCMS) architecture, used in previous studies [2, 7, 10], is not a practical solution to deal with large number of streams. The reason is that, FCMS architecture involves training $2^N - 1$ neural networks in presence of N-streams. For example, 7 sub-band system used in [6, 4] requires $2^7 - 1 = 127$ fusion networks.

In this paper, we propose a practical multistream architecture. Our previous work [5] showed that multiple neural networks used in FCMS architecture can be replaced by a single neural network. This resulted in a significant reduction in the complexity during training phase system. We further simplify the training procedure by using single-stage architecture instead of 2-stage architecture used in [5]. We also propose a technique to reduce the test time complexity of the system. The technique is based on the idea that stream combinations can be organized into a tree, where nodes of the tree represent stream combinations. Advantage of this organization is, we can use tree traversal algorithms to find the best stream combination efficiently. We applied the technique on a 9-stream system. Using this technique, we are able to reduce average number of forward passes from 511 to 30.

Rest of the paper is organized as follows: Sec. 2 describes proposed improvements to multistream architecture. Usefulness of various modules in multistream ASR is illustrated in Sec. 3. Performance in noise robust ASR experiments are presented in Sec. 4. Sec. 5 concludes the paper with a summary of the proposed techniques.

2. Proposed Multistream architecture

2.1. Training stage:

Figure 1 illustrates proposed multistream architecture. For simplicity, we illustrate the architecture for a 2-stream case. The principle can be extended to any number of streams. Input feature vector to the proposed fusion network is formed by concatenating features from individual streams. Just before concatenation of the features, we employ a binary switch (Z_i), which can multiply entire feature vector of stream-i with 0 or 1. During training, each switch (Z_i) acts as an independent Bernoulli random variable. In this work, we use $p = 0.5$ for all the switches. This makes all switch combinations, [1, 1], [1, 0] and [0, 1], equally likely during training. If null-switch combination, [0, 0], is observed, we resample a non-null switch combination ([1, 1], [1, 0] and [0, 1]). During training, value of a switch changes at every frame. So the network can see one of the following input patterns, [stream1, **0**] or [**0**, stream2] or [stream1, stream2] at every frame.

During test, when some of the streams are corrupted, robustness can still be retained by dropping out (i.e. multiplying with zero) feature vectors from corrupted streams. The fusion neural network can break down if it sees a test vector with some

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.

http://dx.doi.org/10.21437/Interspeech.2016-619

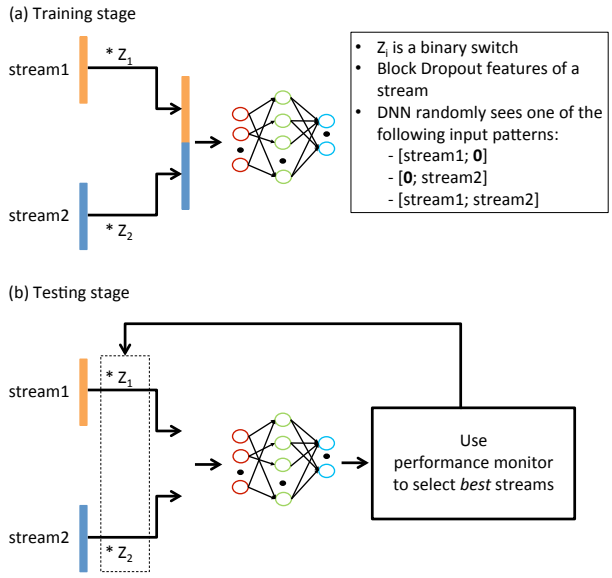

Figure 1: *Training and testing stages of proposed Multistream architecture. Multiple networks in FCMS architecture are replaced by a single network. Complexity of testing stage is also simplified by hierarchical organization stream combinations.*

of its elements as zeros. We avoid this by randomly dropping out features of a stream during training. This forces the network to learn to classify even when part of test vector is zero.

The technique was first proposed in our previous paper [5]. In [5], we used a 2-stage architecture. The first stage involves forming band-limited streams, and training a separate neural network for each stream. The posteriors from the first stage neural network are transformed using TANDEM procedure. TANDEM features from the first stage are concatenated and used as features for second stage neural networks. In this work, we use single stage architecture instead of 2-stage architecture. Acoustic features from each stream are concatenated and used as input to multistream neural network.

2.2. Testing stage:

For a given test utterance, we need to identify stream combination which results in lowest error-rate. In previous works [2, 4, 10], posteriors of all the stream combinations are evaluated using performance monitoring technique. In this approach, number of forward passes required to identify best stream combination increases exponentially with number of streams. For example, a 7-stream system requires 127 forward passes [4]. The large number of forward passes at test time can deter practical applicability of the system.

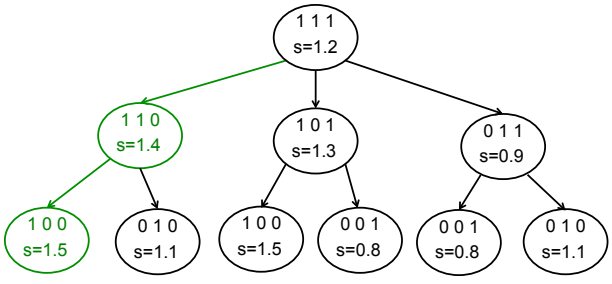

Figure 2: *Illustration of proposed technique to find best stream combination.*

In this section, we propose a technique to reduce the test time complexity of multistream system. The technique is based on the idea that stream combinations can be organized into a tree, where each node represents a stream combination. Figure 2 shows an example organization of stream combinations in a tree. Using this organization, we can use tree traversal algorithms to efficiently search for best stream combination.

Pseudo-code of the search algorithm is described in algorithm 1. Using the algorithm, we reduce the worst-case complexity from $O(2^N)$ to $O(N^2)$ number of evaluations, where N . In practice, number of evaluations required are much fewer than $O(N^2)$. For the 9-stream system used in table 1, number of evaluations required to find best stream combination are $\approx$ 30.

```
1  parent_node = root_node ;
2  parent_score = root_score ;
3  FindBestChildNode(parent_node, parent_score)
4      compute scores of all child nodes ;
5      if parent_score ≥ max(child_scores) then
6          return (parent_node, parent_score) ;
7      else
8          best_child_node = (child_node with score ==
           max(child_scores)) ;
9          FindBestChildNode(best_child_node,
           best_child_node_score) ;
10     end
11 end
```

Algorithm 1: *Search algorithm used to find best stream combination. In our implementation* root_node *refers to combination where all the streams are present.*

3. Controlled experiments

3.1. Subband streams

Log-Mel filterbank features, spanning frequency range of 0-8000 Hz, are computed from the speech signal. In each Mel band, TRAP features are computed by taking DCT transform over a temporal context of 11 frames. The resulting TRAP features are grouped into Bark critical bands. Each stream is set to cover 2 Bark bands. This results in 9 subband streams.

3.2. Experimental setup

The first set of experiments are reported using models trained 15 hour subset of Switchboard-1 Release 2 (LDC97S62), and tested on Hub5 '00 (LDC2002S09) and its variants. More details about the training and test sets, and language models can be found in [14]. Variants of original test set are created by artificially adding different noises at various signal-to-noise ratios. We designed a synthetic band-limited noise which corrupts only few streams (around 3-4 subband streams). The synthetic noise guarantees that at least some of the streams remain uncorrupted by noise, satisfying the basic premise of multistream system. We also used subway, volvo, factory and babble noises from NOISEX [11] database. These noises are used to analyze the behavior of the system in a more natural real world settings.

The ASR system is trained using Kaldi speech recognition toolkit [16]. A GMM-HMM system is trained on speaker adapted MFCC features [14]. The GMM-HMM system is used to generate context-dependent alignments, which are used to train DNN models. The DNN models used in this section consist of 4 hidden layers. Each hidden layer consist of 1500 sigmoidal neurons. The models are trained using cross entropy cost function.

3.3. Performance Monitoring module

During test time, we use performance monitoring score as proxy to accuracy, of a stream combination. We combine autoencoder and M-delta based performance monitor measures to identify stream combination which results in lowest error-rate. We briefly describe these measures in this section. More detailed description can be found in [8].

Autoencoder measure: Application of autoencoders for performance monitoring task is based on the following idea: For a given classifier, the best performing posteriors are its training data. Performance of a test utterance can be estimated by comparing the test posteriors with respect to the model derived on the classifier's training data posteriors. During testing, quality of posteriors of a stream combination is evaluated using reconstruction error values from the autoencoder.

M-delta measure: Main idea behind the measure is as follows: posterior vectors belonging to the same class should have smaller divergence than the divergence between posterior vectors belonging to different classes. M-delta measure is defined as $\mathcal{M}delta = \mathcal{M}^{\mathrm{ac}} - \mathcal{M}^{\mathrm{wc}}$, where $\mathcal{M}^{\mathrm{ac}}$ and $\mathcal{M}^{\mathrm{wc}}$ represent the accumulated KL-divergence computed from a data pair from the same class and that from a data pair from different classes, respectively. $\mathcal{M}^{\mathrm{ac}}$ and $\mathcal{M}^{\mathrm{wc}}$ are estimated from $\mathcal{M}$ measure [13]. For a test utterance, M-delta measure is computed for each stream combination. Posteriors from stream combination having highest M-delta measure are selected.

3.4. Results

Table 1 show comparison results of baseline DNN with proposed multistream DNN. The WER results are on test sets described in 3.2. Baseline DNN refers to a DNN trained in a standard way. Multistream DNN refers to a DNN trained using proposed technique (described in 2.1). Since the training procedures of baseline DNN and multistream DNN differ, we evaluated the performance of multistream DNN when the performance monitor module is not used. Table 1 show that multistream DNN performs better than baseline DNN in all the test conditions. This is due to the block dropout procedure introduced during training of multistream DNN, which helps in regularizing the model.

Next we applied performance monitoring techniques based on autoencoder and MDelta measures. Table 1 shows that application of performance monitoring techniques significantly improves performance in band-limited noisy conditions. This shows that when the noises really localized only to few portions of signal space, significant improvements can be obtained by discarding information from low SNR portions and using only high SNR portions. Even in subway and volvo noises, we observed good improvements by using performance monitoring module. Whereas, in factory and babble noise conditions the improvements are not as substantial as that of band-limited, subway and volvo conditions. This might be due to broad-band nature of babble and factory noises, compared to band-limited, subway and volvo conditions

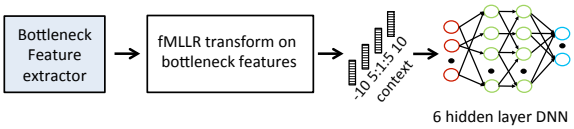

Figure 3: *Block diagram of state-of-the-art ASR system used in the present work. Noise robustness of the system is improved by replacing neural network in Bottleneck Feature extractor module with proposed Multistream neural network.*

Also, the two performance monitoring techniques seems to be competitive in most of the conditions, with MDelta measure being slightly better than autoencoder. Combination of these measures further improved the performance in all the noise conditions, which shows complementary nature of the measures. Next we applied the multistream DNN with performance monitoring module, to improve robustness in noise robust ASR tasks with real noises.

4. Noisy speech recognition experiments

In this section, we apply multistream DNN with performance monitoring module to improve robustness in noisy ASR tasks. The ASR pipeline used for this purpose is shown in figure 3. First component in the pipeline is a neural network trained to extract bottleneck features (BNF) from acoustic features, referred to as bottleneck feature extractor. Bottleneck feature extractor used in present work consist of 3 hidden layers, consisting of 1500 neurons each. These are followed by 40 dimensional linear bottleneck layer and a final logistic regression layer. Inputs to bottleneck feature extractor are TRAP features computed from log-Mel filterbanks (described in Sec. 3.1). The bottleneck feature extractor is trained on context-dependent targets, obtained from a GMM-HMM system (described in Sec. 3.2).

Once the bottleneck feature extractor is trained, the entire training data is forward passed to compute BNFs of the training data. A GMM system is trained on BNFs to estimate fMLLR transforms. A fully connected DNN is trained on the fMLLR+BNF features, using cross entropy (CE) and sequence minimum Bayes risk (sMBR) criterion.

We replace the bottleneck feature extractor with proposed multistream neural network. This network is referred as multistream bottleneck feature extractor. In order to have a fair comparison with baseline system, we keep the architecture of multistream bottleneck feature extractor same as that of baseline bottleneck feature extractor. This results in same number of parameters for both the systems.

Multistream bottleneck feature extractor is trained using the technique described in Sec. 2.1. BNFs for training further stages of the pipeline (GMM and DNN acoustic models on BNFs) are computed using the stream combination where all streams are present. During testing phase, optimal stream combination is identified by using performance monitoring module (Sec. 3.3), and the corresponding BNFs are extracted and given as input to further stages of the pipeline.

4.1. Aurora4 experiments:

We first demonstrate effectiveness in Aurora4 ASR task. Aurora4 task is a small scale, medium vocabulary speech recognition task, aimed at improving noise and channel robustness.

Table 1: *Comparison of WER (%) of proposed Multistream DNN with a feed-forward DNN, on eval2000 (clean) and its noisy variants. All the models are trained on 15 hour subset of Switchboard corpus.*

System \Test condition	clean	band-limited noise	subway		volvo		factory		babble	
			10dB	20dB	10dB	20dB	10dB	20dB	10dB	20dB
Baseline DNN	34.3	64.6	71.3	49.4	79.6	50.5	78.0	48.3	78.2	48.4
Multistream DNN	32.8	51.6	68.8	45.7	77.8	45.6	75.8	44.8	76.6	44.9
+AE_PM	33.0	40.9	62.5	44.5	73.7	44.9	74.4	44.7	75.1	44.9
+MDelta_PM	**32.6**	40.6	59.6	42.8	73.1	44.6	73.6	44.4	73.4	44.5
+AE+MDelta_PM	32.7	**40.4**	**59.3**	**42.7**	**72.4**	**44.3**	**73.4**	**44.1**	**73.3**	**44.5**

The database is based on DARPA Wall Street Journal (WSJ0) corpus which consist of recordings of read speech, with 5000 word vocabulary size. The training set consists of 14 hours of multi-condition data, sampled at 16 kHz. The 14 hours of data is comprised of 7137 utterance from 83 speakers. Half of the utterances were recorded by the primary Sennheiser microphone and the other half were recorded using one of a number of different secondary microphones. Both halves include a combination of clean speech and speech corrupted by one of six different noises (street traffic, train station, car, babble, restaurant, airport) at 10-20 dB signal-to-noise ratio.

The test set consist of 14 conditions, with 330 utterances for each condition. The conditions include clean set recorder with primary Sennheiser microphone, clean set with secondary microphone, 6 additive noise conditions which include airport, babble, car, restaurant, street and train noise at 5-15 dB signal-to-noise ratio (SNR) and 6 conditions with the combination of additive and channel noise.

Table 2 shows the comparison results of baseline system and multistream system. BNF_Xtr rows in table show decoding results at the bottleneck feature extractor stage. Similar to the results in table 1, we observe improvements by using multistream neural network (relative improvement of 18.6 % across all the conditions). The improvement is substantial in secondary microphone conditions (C and D), which illustrates robustness of multistream approach.

In order to compare multistream adaptation and traditional speaker adaptation, we report results obtained from models trained on fMLLR features. DNN_CE rows show decoding results obtained from DNN models trained on fMLLR features, by minimizing cross entropy objective function. We can observe from the table that, DNN models in multistream system are consistently performing better than DNN models in baseline system. This result shows that robustness obtained by using multistream architecture can be complementary to traditional speaker adaptation techniques. The gains are consistent (13.4 % relative improvement) even in state-of-the-art DNN models trained on sequence minimum Bayes risk criterion (DNN_sMBR rows in the table).

4.2. IARPA Babel experiments

In this section, we report improvements obtained by using multistream system in IARPA Babel data [12]. Experiments are performed on BABEL OP3 languages: Igbo (IGB), Javanese (JAV), Guarani (GAU), Amharic (AMH) and Mongolian (MON). We use Full Language Pack (FLP) condition to train acoustic models of the ASR pipeline (Fig. 3). FLP scenario has $\approx$ 46 hours of transcribed audio available for each language. Results are reported using Kneser-Ney smoothed tri-gram language models.

Table 3 shows the results for the 5 languages. It is evident from the table, that multistream system is performing better than baseline system. We observed a 0.7 % absolute decrease in WER, in 4 of the 5 languages. The reason for this improvement might be due to various real world acoustic conditions present in Babel data [12]. These results show the generality of the proposed system.

Table 2: *Comparison of WERs (%) at various stages in the ASR pipeline of Fig. 3. Proposed Multistream system consistently outperforms Baseline system at all stages of the pipeline.*

		A	B	C	D	Avg.
Baseline	BNF_Xtr.	4.17	7.80	12.42	21.85	13.89
	DNN_CE	3.16	5.10	5.70	16.33	9.82
	DNN_sMBR	3.36	5.01	5.70	16.10	9.70
Multistream	BNF_Xtr.	4.26	7.12	7.27	17.34	11.30
	DNN_CE	2.43	4.67	3.77	14.24	8.55
	DNN_sMBR	2.43	4.55	3.72	14.04	8.41

Table 3: *WERs (%) of Monolingual BABEL systems trained using Full language pack data.*

System	IGB	JAV	GAU	AMH	MON
Baseline	60.5	58.0	46.7	43.6	52.2
Multistream	59.7	57.3	46.1	43.5	51.5

5. Conclusions

In this paper, we proposed a framework to make multistream architecture more practical. We significantly reduced complexity during training of a multistream system. We also made testing phase much faster by using an efficient search technique to identify best stream combination. The proposed techniques can be used to construct a multistream system with much larger number of streams. We applied multistream neural network with proposed training and testing techniques, in various noisy speech recognition tasks. Noticeable improvements were observed over state-of-the-art baseline ASR architecture. Also, the results indicate that proposed system is more robust to acoustic mis-matches than baseline system.

6. References

[1] H. Hermansky, "Multistream recognition of speech: Dealing with unknown unknowns," Proc. *IEEE*, vol. 101, no. 5, pp. 1076-1088, May 2013.

[2] 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.

[3] E. Variani, F. Li, and H. Hermansky "Multi-Stream Recognition of Noisy Speech with Performance Monitoring," Interspeech 2013.

[4] E. Variani and H. Hermansky. "Estimating Classifier Performance in Unknown Noise," Interspeech 2012

[5] S. H. Mallidi and H. Hermansky, "Novel neural network based fusion for Multistream ASR," in Proc. ICASSP 2016.

[6] H. Hermansky, S. Tibrewala, and M. Pavel, "Towards ASR on partially corrupted speech," in *Proc. Int. Conf. Spoken Lang. Process.*, 1996, pp. 462-465.

[7] E. Variani, F. Li, and H. Hermansky "Multi-Stream Recognition of Noisy Speech with Performance Monitoring," Interspeech 2013.

[8] S. H. Mallidi, T. Ogawa and H. Hermansky, "Uncertainty estimation of DNN classifiers," 2015 IEEE ASRU, Scottsdale, AZ, 2015, pp. 283-288

[9] S. H. Mallidi, T Ogawa, K Vesely, P S Nidadavolu, and H. Hermansky "Autoencoder based multi-stream combination for noise robust speech recognition", Interspeech, 2015.

[10] H. Misra, H. Bourlard, and V. Tyagi. "New entropy based multi-stream combination." In Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). IEEE International Conference on, vol.1, pp. I-193. IEEE, 2004.

[11] A. Varga and H. J. M. Steeneken, "Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Commun.*, vol. 12, no. 3, pp. 247-251, 1993.

[12] IARPA Babel Program, http://www.iarpa.gov/index.php/research-programs/babel, accessed: 2015-02-27.

[13] Hermansky, Hynek, Ehsan Variani, and Vijayaditya Peddinti. "Mean temporal distance: Predicting ASR error from temporal properties of speech signal".Acoustics, Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

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

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

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