The THUEE Systems for the IARPA OpenASR20 Evaluation

Jing Zhao, Guixin Shi, Guan-Bo Wang, Wei-Qiang Zhang
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
wqzhang@tsinghua.edu.cn

***Abstract*—This paper introduces the systems of THUEE for the IARPA Open Automatic Speech Recognition Challenge (OpenASR20). We compete in the constrained training condition only. We adopt the hybrid NN-HMM acoustic model and an N-gram Language Model (LM) to construct our basic Automatic Speech Recognition (ASR) systems. The acoustic model is proposed as CNN-TDNN-F-A, which combines Convolution Neural Network (CNN), Factored Time Delay Neural Network (TDNN-F) and self-attention mechanism. As for low-resource condition, we apply speed perturbation, SpecAugment, WavAugment as well as reverberation to the original speech data as data enhancement. We also clean up the original data to filter interference information. Besides, we train the LMs with some external text data from train set of IARPA Babel program. System fusion is conducted by ROVER.**

***Keywords*— low-resource languages, data augmentation, CNN-TDNN-F-A acoustic model, system fusion**

I. Introduction

The goal of the OpenASR20 is to assess the state of the art of Automatic Speech Recognition (ASR) technologies for low-resource languages. For most of the languages in the world, there are no applicable ASR systems because of the lack of high-quality annotation speech data. It is challenging to built an strong ASR system with limited speech data, script texts as well as lexicons.

We describes our ASR systems in detail to show the whole procedure that we deal with the challenges. For our hybrid acoustic model, we propose the CNN-TDNN-F-A network as the essential part, which is trained with lattice-free maximum mutual information (LF-MMI) criterion [1]. The model introduces self-attention mechanism [2] to the combination of CNN and TDNN-F [3] in order to learn more positional information from the input.

Since the major challenge is low-resource condition, we combine some different kinds of data augmentation methods to get additive improvement, such as speed perturbation [4], volume perturbation, SpecAugment [5], WavAugment [6] as well as reverberation. They are effective to the ASR performance especially under low-resource condition.

Besides, systems' diversity is important for the final fusion. We have trained more than three systems of each language to make further use of the differences of the single systems by system fusion.

II. Workflow

We perform the experiments with Kaldi speech recognition toolkit [7]. Since NN-HMM acoustic model has promising performances for ASR, we develop our systems with the hybrid structure. We display the main workflow in Fig. 1, which consists of pre-processing, data augmentation, training, decoding and system fusion roughly.

First a Gaussian Mixture Model (GMM) is trained by several training and force-aligning iterations, which includes Speaker Adaptation Trainings (SAT). Data cleanup can be applied to the original speech and text data with the trained GMMs. Then, some augmentation methods can directly process the raw audio to add diversity and enlarge the quantity of training data, such as speed perturbation [4] and reverberation. We obtain three or six times of the original quantity of data for the following training. High-resolution MFCC feature and pitch feature are extracted from the augmented data.

Besides, SpecAugment [5] is applied to the combined acoustic features to augment data further. When training the CNN-TDNN-F-A acoustic model, we also add i-vectors along with the MFCC and pitch features to integrate speaker information into the model. As for language model, we build a N-gram LM by SRILM [8] with some extra text data from IARPA Babel program [9]. In addition, a Recurrent Neural Network (RNN) LM is also used to rescore lattices after decoding. Finally, several different system outputs are fused by ROVER [10] to obtain a better performance. The details are described in the following sections.

III. Acoustic Model

A. *CNN-TDNN-F-A architecture*

We propose the CNN-TDNN-F-A network as the neural network acoustic model, which combines Convolution Neural Network (CNN), Factored Time Delay Neural Network (TDNN-F) [3] and self-attention mechanism. The architecture is displayed in Fig. 2. The popular TDNN-F networks are the basic part of our acoustic model, which is structurally the same as a TDNN whose layers have been compressed via SVD, but is trained from a random start with one of the two factors of each matrix constrained to be semi-orthogonal. A regular TDNN-F block consists of a linear layer, an affine component,

The corresponding author is Wei-Qiang Zhang.

This work was supported by the National Natural Science Foundation of China under Grant No.U1836219.

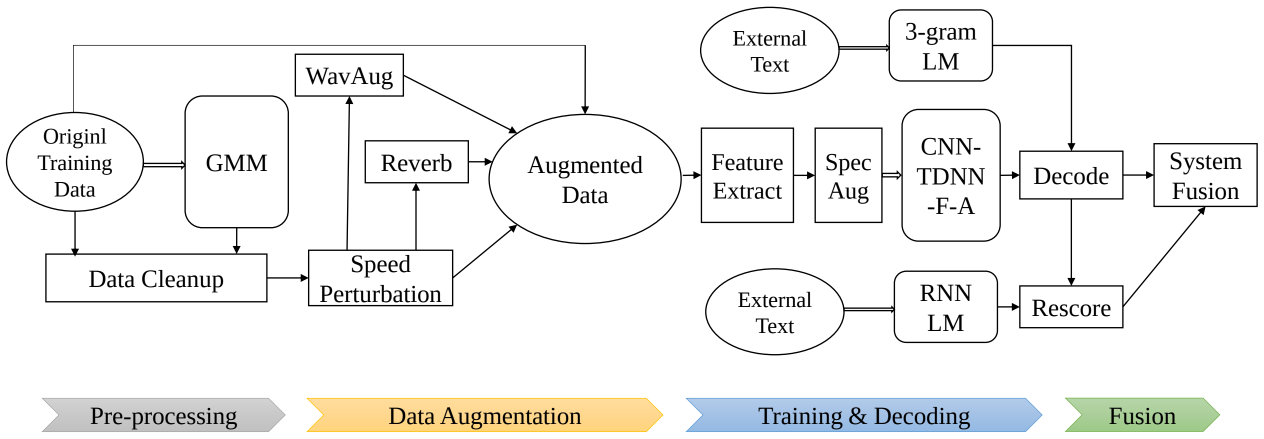

Fig. 1. Workflow of the ASR system.
The whole system process can be roughly divided into pre-processing, data augmentation, training, decoding and system fusion.

a ReLU nonlinearity component, and batch normalization operation followed by dropout. The CNN-TDNN-F-A network contains 11 TDNN-F blocks in total with the hidden dimension of 768 for the first 9 blocks, 760 for the last two layers to match the attention layer, and a bottleneck dimension of 160 same for all the layers. For different system settings, the bottleneck dimension is also set to 120 or 256. All the TDNN-F layers except for the first layer connected to the CNN component have a time stride of 3.

CNN has been applied to the speech recognition task successfully by introducing three extra concepts over the simple fully connected feed-forward NN: local filters, max-pooling, and weight sharing [11]. Previous experiments have showed the efficiency of CNN-TDNN [12], and we replace the TDNN with TDNN-F which performs better especially with small amount of data. In our architecture, the convolution block is obtained by a convolutional layer and a ReLU nonlinearity component followed by batch normalization. We adopt 6 convolution blocks at the beginning of the acoustic model with concatenation of i-vectors and MFCCs as input. The number of filters is 48, 48, 64, 64, 64, 128 successively.

Recently the self-attention layer has been successful with multi-head attention which allows the networks to jointly attend to information from different representation subspaces at different positions [13]. Besides, In [2] the self-attention layer was adopted in a time-restricted fashion, which is more suitable for speech recognition. We combine the self-attention mechanism with the CNN-TDNN-F, and thus obtain our final CNN-TDNN-F-A architecture. The self-attention model is composed of an affine component, an attention nonlinearity component, and a ReLU nonlinearity component followed by batch normalization. The location of the layer should be close to the end of the network to obtain better performance. And we set the self-attention block the third layer from the bottom. In detail, the multi-head attention component has 20 attention heads along with a key-dimension of 8 and a value-dimension of 16. The context is set to $[15; 6]$. The settings are adjusted to fit for the network dimension 768 according to some conclusions in [2]. For example, a key to value ratio of 0.5 about is slightly better than other ratios.

B. Training Settings

Following the hybrid system training pipeline, we first train some GMM-HMM models, which use GMM to model the HMM output probability density, to produce high-quality alignments to force-align the training dataset for the NN-based acoustic model. We use Perceptual Linear Prediction (PLP) feature with pitch feature to train the GMM-HMM model. The iterative training process including modeling monophone, creating triphone model, applying Linear Discriminant Analysis (LDA) and Maximum Likelihood Linear Transform (MLLT), performing speaker adaptive training.

The NN acoustic models of our systems are based on high-resolution MFCC feature with pitch feature for some of the ten languages,such as Cantonese, Kurmanji-Kurdish and Somali. For the rest languages, only MFCC features are used. We also use i-vector features for speaker adaptation. I-vectors are fixed-length vectors containing speakers information, and have become a common technique for speaker recognition. In our systems, we train the i-vectors based on a diagonal UBM for speaker adaptation [14]. In order to adapt to the CNN structure, the extracted 100-dim i-vectors are mapped to 200-dim by linear transformation before concatenating with MFCCs.

For training procedure, the CNN-TDNN-F-A acoustic model is trained with chain/chain2 component of Kaldi toolkit which adopts LF-MMI criterion [1]. The batch size is set to 128 or 64 with 6 epochs training in total. The initial learning rate is 0.0005 and the final learning rate is 0.00005.

IV. Language Model

We train the 3-gram language models by SRILM [8], with some extra text data from IARPA Babel program [9] in addition to transcripts of the provided training data. Below

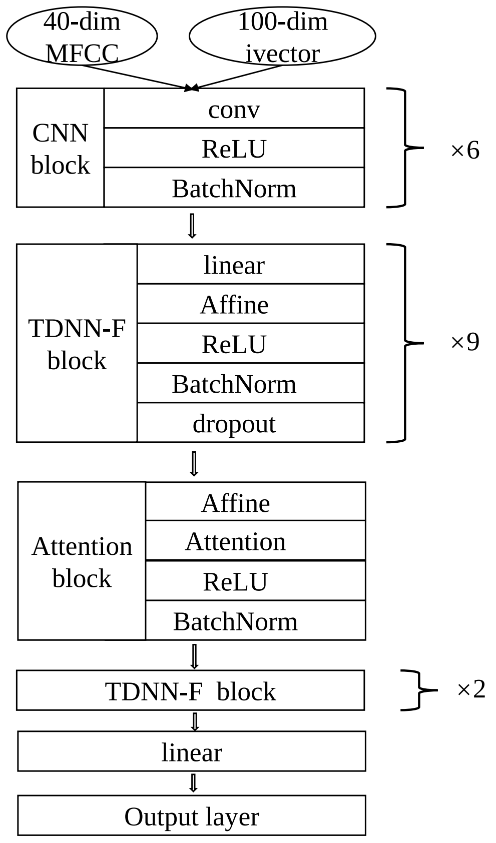

Fig. 2. CNN-TDNN-F-A architecture

is the related dataset list in IARPA Babel program, and only transcripts of the training part are used.

- IARPA-babel101b-v0.4c-build
- IARPA-babel104b-v0.bY-build
- IARPA-babel107b-v0.7-build
- IARPA-babel204b-v1.1b-build
- IARPA-babel205b-v1.0a-build
- IARPA-babel305b-v1.0c-build
- IARPA-babel307b-v1.0b-build
- IARPA-babel401b-v2.0b-build
- IARPA-babel402b-v1.0b-build

Besides, we also employ lattice rescoring with NN-based LMs, which are composed by several TDNN-LSTM networks [15]. The training data for the NN-based LMs are the same as above.

V. Data Processing

A. Cleanup

We perform data cleanup to remove bad portions of the training transcripts and do other minor modifications of transcripts such as allowing repetitions for disfluencies, and adding or removing non-scored words. The cleanup relies on the GMM-based models we have trained. Specifically, the SAT model is employed here.

The operation is more effective to the unclean speech data. Most languages' performance are improved by cleanup, except for Mongolian, Tamil and Vietnamese, according to the decoding results of the 10-hour development set. Furthermore, a more strongly biased LM usually brings better performance.

B. Augmentation

Since the major challenge is to deal with low-resource condition, it's essential to adopt appropriate methods of data augmentation. In order to make full use of the limited training data, we adopt several popular techniques at the same time to enhance the robustness of our ASR systems and make our system more invariant to properties of the evaluation data.

1) Speed Perturbation: In [4], speed perturbation is proposed as an effective data augmentation method by processing the raw signal. We adopt the method to change the speed of the training audio signal, producing 3 versions of the original signal with speed factors of 0.9, 1.0 and 1.1, which is beneficial to avoid overfitting and improve robustness of the models. By speed perturbation, we increase the data quantity by three times.

2) Volume Perturbation: Volume perturbation is adopted after speed perturbation, which is also conducted on the raw speech audio. It improves the volume robustness of the model.

3) SepcAugment: We apply SpecAugment [5] to the MFCC features, along with pitch feature for some languages, before input to the acoustic neural networks. The policy consists of warping the features, masking blocks of frequency channels, and masking blocks of time steps.

4) WavAug: The recently proposed Wavaug is a time-domain data augmentation library, which integrates 5 augmentations: pitch modification, additive noise, reverberation, band reject filtering, and time masking [6]. Since most of the augmentations have already been implemented by the methods mentioned above, we only adopt reverberation as a supplementary method. We add small-room, medium-room and large-room effect reverberation with same weight to generate a reverberated copy of the original speech data. We can obtain 6 times quantity of original data based on the speed perturbation augmented results.

VI. Pre-and-Post processing

A. Speech Activity Detection

For the evaluation period, Speech Activity Detection (SAD) is a necessary operation to segment the audio appropriately so that we can decrease loss of useful speech clips and improve the decoding efficiency and accuracy. In our SAD component, we combine a convolutional recurrent neural network (CRNN) and a recurrent neural network (RNN) to make the system more robust. In addition, we add a speech-enhancement module and a one-dimensional dilation-erosion module to our SAD system. The SAD system follows work in [16]. For each audio input, firstly, we preprocess it and extract the fbank features. The system will output speech existence in every frame separately. Finally, the output passes the post-processing module and becomes the final output.

B. Decoding

In our systems, we use a WFST-based method for decoding based on Kaldi Toolkit. For the firstpass decoding, we simply use the N-gram model as the decoding language model. The decoding beam is set to 15.0 while the beam used in lattice generation is 8.0. The LM weight is chosen in integers from 8 to 12. Besides, a two-layer LSTM language model is trained for lattice rescoring [15].

C. System Fusion

In order to enhance the diversity of our systems, we use some setups with different data augmentations or different network dimensions. After we obtain the recognition results of each system, we adopt ROVER method [10], which is a post-recognition process which models the output generated by multiple ASR systems as independent knowledge sources that can be combined and used to generate an output with reduced error rate.

D. Results Filtering

Finally we obtain the ASR results from lattice, we filter the words lists by the corresponding degree of confidence. The threshold is set to 0.3, which means the decoding results with confidence value below 0.3 would be abandoned. The operation is effective to reduce the insertion error.

VII. Results

A. Development Set

Our systems' performance of the ten languages under constrained condition on the development set is shown in Tab. I. The WERs are obtained by sclite [17] with Reference File Format (STM) generated by ourselves, which has minor differences from the NIST OpenASR scoring server. Besides, the results are directly obtained by firstpass decoding, without RNNLM rescoring, system fusion or post-filtering, which we have all applied to the evaluation set. It is not surprising that the results we show in Tab. I on the development set is worse than the final results on the evaluation set in Tab. II.

The results in Tab. I are obtained by the same acoustic model, CNN-TDNN-F-A architecture, except for minor differences in the bottleneck dimension of TDNN-F layers. The three different bottleneck dimensions have little effect on the results directly but can make contribution to the fusion systems. Besides, we also have some ablation experiments to determine the specific acoustic model architecture, for example, the number of TDNN-F layers, the dimension of networks, the location and other detailed settings of the self-attention layer. According to the results of some languages, the final setting introduced in Sec. III-A is a better choice. We don't present these raw results since they are too detailed and redundant. As for the self-attention layer, it doesn't bring as obvious improvement as we hope actually. We think the limited data can be a main reason.

TABLE I
WER of ASR systems on DEV set

Language	System	WER
Amharic	cnn_tdnnfa_chain2_sp	0.490
	cnn_tdnnfa_cleanup_chain2_sp	0.487
	cnn_tdnnfa_bn256_chain2_sp	0.493
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.490
Cantonese	cnn_tdnnfa_pitch_chain2_sp	0.492
	cnn_tdnnfa_pitch_cleanup_chain2_sp	0.483
	cnn_tdnnfa_cleaup_bn256_chain2_sp	0.488
	cnn_tdnnfa_bn120_chain2_sp	0.484
Guarani	cnn_tdnnfa_chain2_sp	0.502
	cnn_tdnnfa_cleanup_chain2_sp	0.500
	cnn_tdnnfa_rvb_chain_sp	0.499
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.503
Javanese	cnn_tdnnfa_chain2_sp	0.578
	cnn_tdnnfa_cleanup_chain2_sp	0.575
	cnn_tdnnfa_rvb_chain_sp	0.571
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.577
Kurmanji-Kurdish	cnn_tdnnfa_chain2_sp	0.673
	cnn_tdnnfa_cleanup_chain2_sp	0.671
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.673
Mongolian	cnn_tdnnfa_chain2_sp	0.524
	cnn_tdnnfa_bn256_chain2_sp	0.525
	cnn_tdnnfa_rvb_chain_sp	0.524
	cnn_tdnnfa_bn120_chain2_sp	0.524
Pashto	cnn_tdnnfa_chain2_sp	0.498
	cnn_tdnnfa_cleanup_chain2_sp	0.496
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.496
Somali	cnn_tdnnfa_chain2_sp	0.596
	cnn_tdnnfa_cleanup_chain2_sp	0.592
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.594
Tamil	cnn_tdnnfa_chain2_sp	0.691
	cnn_tdnnfa_cleanup_chain2_sp	0.695
	cnn_tdnnfa_cleanup_bn256_chain2_sp	0.694
	cnn_tdnnfa_bn120_chain2_sp	0.692
Vietnamese	cnn_tdnnfa_chain2_sp	0.488
	cnn_tdnnfa_cleanup_chain2_sp	0.490
	cnn_tdnnfa_cleanup_bn120_chain2_sp	0.493

B. Evaluation Set

Our systems' performance of the ten languages under constrained condition on the evaluation set is shown in Tab.II, which are released by NIST OpenASR scoring server. For each language, the submissions are composed by 2 or 3 single systems and fusion systems which shows better performance.

VIII. Hardware and Time Description

The hardware of our proposed system is shown in Tab.III. As for the required time, for each language, the elapsed wall clock time is approximately 3 hours for a single whole system, which can be divided into 3 main stages, 40 minutes for GMM training, 2 hours for acoustic model training and 20 minutes for decoding approximately. GPU resources are only used for NN acoustic model training. The corresponding total CPU time is about 20 hours since the number of threads is usually set to 16, and the total GPU time is 2 hours or so.

TABLE II
WER OF ASR SYSTEMS ON EVAL SET

The submission names omits the same prefix
"OpenSAT-2020_OPENASR20_evaluation_SYS-00166_THUEE"

Language	submission	WER
Amharic	20201109-022700-2243	0.462
	20201109-023132-6706	0.464
	20201109-083551-2967	0.581
	20201109-083655-5658	**0.458**
Cantonese	20201109-083842-7233	0.453
	20201109-083946-5041	0.449
	20201109-222733-6488	0.437
	20201109-222814-0301	**0.436**
Guarani	20201109-084205-2582	0.490
	20201109-084304-0489	0.485
	20201109-084408-3474	0.480
	20201109-222854-4599	0.465
	20201109-222927-2020	**0.461**
Javanese	20201109-084519-8636	0.556
	20201109-084658-9709	0.545
	20201109-090736-7503	0.545
	20201109-223103-6921	0.522
	20201109-223139-8894	**0.521**
Kurmanji-Kurdish	20201109-090842-2972	0.697
	20201109-091002-1182	0.686
	20201109-091125-6511	0.686
	20201109-223238-8497	0.670
	20201109-223316-8019	**0.669**
Mongolian	20201109-091253-0860	0.491
	20201109-091413-8127	0.481
	20201109-223440-4435	0.463
	20201109-223514-4223	**0.454**
	20201109-091528-6459	0.505
Pashto	20201109-091528-6459	0.505
	20201109-091607-2789	0.503
	20201109-091646-8853	0.513
	20201109-223549-6312	0.490
	20201109-223614-6704	**0.486**
Somali	20201109-091749-7946	0.613
	20201109-091829-1026	0.609
	20201109-091910-4433	0.600
	20201109-223652-7598	0.606
	20201109-223736-2987	**0.596**
Tamil	20201109-092003-2999	0.679
	20201109-092034-0677	0.707
	20201109-092105-4573	0.675
	20201109-223758-5768	**0.660**
	20201109-223829-3558	0.681
Vietnamese	20201109-092210-2051	0.482
	20201109-092319-0150	0.478
	20201109-092408-0119	0.488
	20201109-224120-1204	0.464
	20201109-224158-8184	**0.460**

REFERENCES

[1] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Interspeech*. San Francisco, CA, USA: ISCA, Sep 2016, pp. 2751–2755.

TABLE III
HARDWARE DESCRIPTION

OS	CentOS 7.4 64-bit
CPU num	2
CPU description	28,Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz 112,Intel(R) Xeon(R) CPU E5-4650 v4 @ 2.20GHz
GPU num	1
GPU description	Tesla P100 SMX2 16GB
RAM	256GB
RAM per CPU	128GB
Disk storage	About 1TB

[2] D. Povey, H. Hadian, P. Ghahremani *et al.*, "A time-restricted self-attention layer for ASR," in *proc. ICASSP*. Calgary, AB, Canada: IEEE, Apr. 2018, pp. 5874–5878.

[3] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks." in *Interspeech*, 2018, pp. 3743–3747.

[4] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *INTERSPEECH*. Dresden, Germany: ISCA, Sep 2015, pp. 3586–3589.

[5] D. S. Park, W. Chan, Y. Zhang *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *proc. interspeech*, Graz, Austria, Sep. 2019, pp. 2613–2617.

[6] E. Kharitonov, M. Rivière, G. Synnaeve, L. Wolf, P. Mazaré, M. Douze, and E. Dupoux, "Data augmenting contrastive learning of speech representations in the time domain," *CoRR*, vol. abs/2007.00991, 2020.

[7] P. Daniel, G. Arnab, B. Gilles, B. Lukáš, G. Ondrej, G. Nagendra *et al.*, "The kaldi speech recognition toolkit," *Workshop ASRU*, Jan. 2011.

[8] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *proc. ICSLP - interspeech*, Denver, Colorado, USA, Sep. 2002.

[9] (2011) Babel program. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/babel

[10] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*. IEEE, 1997, pp. 347–354.

[11] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *ICASSP*. Kyoto, Japan: IEEE, Mar 2012, pp. 4277–4280.

[12] A. Georgescu, H. Cucu, and C. Burileanu, "Kaldi-based DNN architectures for speech recognition in romanian," in *SpeD*. Timisoara, Romania: IEEE, Oct 2019, pp. 1–6.

[13] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: NIPS*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[14] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 55–59.

[15] X. Liu, Y. Wang, X. Chen, M. J. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4908–4912.

[16] G.-B. Wang and W.-Q. Zhang, "A fusion model for robust voice activity detection," in *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2019, pp. 1–5.

[17] (2018) SCTK, the NIST scoring toolkit. [Online]. Available: https://github.com/usnistgov/SCTK