The THUEE System Description for the IARPA OpenASR21 Challenge

Jing Zhao, Haoyu Wang, Jinpeng Li, Shuzhou Chai, Guan-Bo Wang, Guoguo Chen, 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 (OpenASR21). We compete in two training conditions, Constrained and Constrained-plus. For Constrained Training Condition, we adopt the hybrid NN-HMM acoustic model to construct our basic Automatic Speech Recognition (ASR) systems. To alleviate the problem of Out-Of-Vocabulary (OOV), we use grapheme-to-phoneme (G2P) method and lexicon extending to make the pronunciation lexicons more comprehensive. The acoustic model CNN-TDNN-F and CNN-TDNN-F-A, which adopts self-attention mechanism are utilized. As for low-resource condition, we apply speed perturbation, SpecAugment, noise addition as well as reverberation to the original speech data as data enhancement. Besides, we train the Languages Model (LM) with some external text data from Build set of IARPA Babel program. Lattice rescoring with RNNLM is applied as well. System fusion is conducted by lattice combination and ROVER. For Constrained-plus Training Condition, we adopts the self-supervised learning framework wav2vec2.0. The released pre-trained model XLSR-53 is employed by different fine-tuning methods. It is helpful to make use of the extra speech data in Babel program in an unsupervised way. During inference, we try to segment the audio with different Speech Activity Detection (SAD) methods, which are complementary at system fusion.**

***Keywords*— low-resource languages, OpenASR2021, lexicon extending, data augmentation, wav2vec2.0**

I. Introduction

The OpenASR21 Challenge is the third open challenge created out of the Intelligence Advanced Research Projects Activity (IARPA) Machine Translation for English Retrieval of Information in Any Language (MATERIAL) program[1]. The goal of the OpenASR21 is to assess the state of the art of Automatic Speech Recognition (ASR) technologies for low-resource languages. Different from OpenASR20, five new languages are added except the ten languages, and the Constrained-plus Training Condition is offered as a new track. Three of the new languages feature additional evaluation datasets for which the output will be scored using case-sensitive criteria. 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. The capabilities tested in the open challenges are expected to ultimately support the MATERIAL task of effective triage and analysis of large volumes of data in a variety of less-studied languages[2].

We describe our ASR systems in detail to show the whole procedure for the Constrained and Constrained-plus Training Condition that we deal with the challenges respectively.

For Constrained Training Condition, we concentrate on hybrid acoustic models with CNN-TDNN-F and CNN-TDNN-F-A network as the essential part, which are trained with Lattice-Free Maximum Mutual Information (LF-MMI) criterion [1]. The latter introduces self-attention mechanism [2] to the combination of CNN and TDNN-F [3] in order to learn more positional information from the input. Out-Of-Vocabulary (OOV) problem is a significant challenge under low-resource condition so that we apply grapheme-to-phoneme (G2P) as well as lexicon extending to obtain more comprehensive pronunciation lexicons. We combine some different kinds of data augmentation methods to get additive improvement, such as speed perturbation [4], volume perturbation, SpecAugment [5] and noise addition with MUSAN [6]. They are effective to the ASR performance especially under low-resource condition. Lattice rescoring with RNNLM are performed forward and backward successively [7]. Besides, systems' diversity is important for the final fusion. We have trained three systems for each language to make further use of the differences of the single systems by system fusion.

For Constrained-plus Training Condition, we obtain evident improvements with extra unlabeled speech data from Babel program[3] and unsupervised pre-trained model of wav2vec2.0 [8] compared to Constrained Training Condition. We utilize two fine-tuning methods to build our systems. Multiple systems trained with training transcripts dealt by different rules and inference results with different Speech Activity Detection (SAD) methods are useful at fusion phase. Besides, the systems trained in Constrained Training Condition can play a role as well in fusion systems.

We briefly introduce our workflows in Sec.II. The systems for Constrained and Constrained-plus Training Condition are described detailed in Sec.III and Sec.IV respectively. Then we show the final results of the EVAL datasets in Sec.V. Finally, hardware and time information are given in Sec.VI.

*Corresponding author

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

[1]https://www.iarpa.gov/index.php/research-programs/material

[2]https://www.nist.gov/document/openasr21-challenge-evaluation-plan

[3]https://www.iarpa.gov/index.php/research-programs/babel

II. Workflow

A. Constrained Training Condition

We perform the experiments with Kaldi speech recognition toolkit [9]. Since NN-HMM acoustic model usually has more promising performances for low-resource ASR than end-to-end model, we develop our systems with the hybrid structure. The brief workflow is showed in Fig.1, which consists of pre-processing, training, decoding and system fusion roughly.

Regarding to the lexicons, we extend the pronunciation lexicons from IARPA Babel program [10] instead of the provided ones. G2P is used to deal with OOV words additionally for the three languages not in Babel program.

Then, the Gaussian Mixture Models (GMM) are trained by several training and force-aligning iterations. 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. The MUSAN dataset [6] is uesd for augmentation. High-resolution MFCC features and pitch features are extracted from the augmented data. Besides, SpecAugment [5] is applied to the combined acoustic features to augment data further. When training the acoustic model, we also add i-vectors along with the acoustic features to integrate speaker information into the model.

As for language model, we build a N-gram LM by SRILM [11] with some extra text data. In addition, a Recurrent Neural Network (RNN) LM is also used to rescore lattices after decoding. Finally, several different system outputs are fused by lattice combination method [12] or ROVER [13] to obtain a better performance. The details are described in the following sections.

B. Constrained-plus Training Condition

The fairseq toolkit[4] is mainly used under Constrained-plus Training Condition. We apply the open-sourced self-supervised pre-trained wav2vec2.0 [8] model to our systems. For model compatibility, all the speech data provided are up or down sampled to 16k Hz.

As for the modeling units during fine-tuning the model, we use characters or graphemes directly in the transcripts. The model is fine-tuned by two kinds of texts with little differences, that are full texts and text with OOV words filtered.

Futhermore, we train the pre-trained wav2vec2.0 model with the speech data of the corresponding language in Babel program in the unsupervised way before fine-tuning. System fusion with different SAD segments are helpful. The results from Constrained Training Condition can make contributions as well.

III. Constrained Training Condition

A. Lexicon

As mentioned above, most of languages in the challenge are from Babel Program. The corresponding pronunciation lexicons in Babel program can cover almost all the words in the train and development set texts. However, for the languages from MATERIAL program and the languages using Case-Sensitive-Scoring criteria, the Babel lexicons are not available and the OOV rate of these languages can reach up to 20 percent with the lexicons provided. To solve the problem caused by OOV words, we use Language G2P models to generate approximate pronunciations for the OOV words of these languages. We use G2P models from LanguageNet [14] which provides Phonetisaurus [15] FST G2P models of around 150 languages, to generate OOV words appeared in train and development set and append them to the original pronunciation lexicons.

We perform lexicon expansion for all the languages to relieve the effects brought by OOV words further. The lexicon extension procedure generates pronunciations and probabilities for the generated words so that we can assign probabilistic quality to ”unknown words” in the language model. First, we treat all entries in the original lexicon as sentences, where each syllable corresponds to a symbol. We train the 3-gram language models by SRILM [11], with some extra text data from IARPA Babel program [10] in addition to transcripts of the provided training data. Then, we generate 12 million sentences from this language model, and for each unique sentence in the generated sentences, we compute its language model probability. The sentences corresponding to the words in the original lexicon are excluded and only the best 1 million sentences remain. Next, we use G2P to get the top few likely spellings corresponding to each generated pronunciation, while recording the probability of each spelling. Finally, we expand the original lexicon with 1 million new entries.

B. Acoustic Model

1) Architecture: The CNN-TDNN-F network is adopted as the neural network acoustic model, which combines Convolution Neural Network (CNN), Factored Time Delay Neural Network (TDNN-F) [3]. We also employ CNN-TDNN-F-A architecture as the same time that combines self-attention mechanism [16]. In [2] the self-attention layer was adopted in a time-restricted fashion, which is more suitable for speech recognition. 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, a ReLU nonlinearity component, and batch normalization operation followed by dropout.

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 [17]. Previous experiments have showed the efficiency of CNN-TDNN [18]. 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. For some languages, such as Cantonese and Tamil, the

[4]https://github.com/pytorch/fairseq

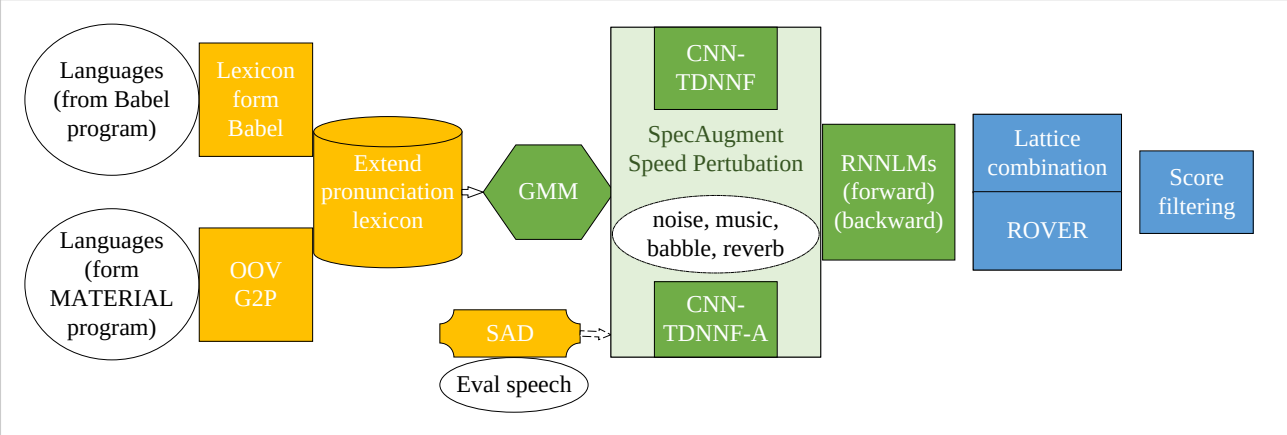

Fig. 1. Workflow of the ASR system.
The main system process can be roughly divided into pre-processing (yellow), training and decoding (green) and post-processes such as system fusion (blue).

pitch features are appended as well. The number of filters is 48, 48, 64, 64, 64, 128 successively.

The AM networks contain 11 TDNN-F blocks in total with the hidden dimension of 768 and a bottleneck dimension of 160 same for all the layers. All the TDNN-F layers except for the first layer connected to the CNN component have a time stride of 3. As for CNN-TDNN-F-A, 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 and 6 for left and right direction respectively. 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 usually better than other ratios.

2) Training Settings: Following the hybrid system training pipeline, we first train the 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 languages,such as Cantonese Georgian, Tagalog and Tamil. 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 [19]. 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 acoustic models are trained with chain/chain2 component of Kaldi toolkit which adopts LF-MMI criterion [1]. The batch size is set to 128 with 6 epochs training in total. The initial learning rate is 0.001 and decays to 0.0001 finally.

C. Language Model

We train the 3-gram language models with kn discount by SRILM [11], with some extra text data from IARPA Babel program [10] in addition to transcripts of the provided training data. Below is the related datasets in IARPA Babel program, and only transcripts of the training part are used. For the languages not in Babel program, only the texts from the 10 hours training set provided is used to train the Language models.

TABLE I
THE EXTRA USED TEXTS IN IARPA BABEL PROGRAM.

Language	Version	#words
Amharic	IARPA-babel307b-v1.0b-build	281k
Cantonese	IARPA-babel101b-v0.4c-build	892k
Guarani	IARPA-babel305b-v1.0c-build	311k
Javanese	IARPA-babel402b-v1.0b-build	309k
Kazakh	IARPA-babel302b-v1.0a-build	270k
Kurmanji	IARPA-babel205b-v1.0a-build	346k
Mongolian	IARPA-babel401b-v2.0b-build	403k
Pashto	IARPA-babel104b-v0.bY-build	888k
Swahili	IARPA-babel202b-v1.0d-build	287k
Tagalog	IARPA-babel106-v0.2g-build	595k
Tamil	IARPA-babel204b-v1.1b-build	486k
Vietnamese	IARPA-babel107b-v0.7-build	923k

Besides, we also employ lattice rescoring with NN-based LMs, which are composed by several TDNN-LSTM networks [7]. The training data for the NN-based LMs are the same as above. Furthermore, we use the same network trained on the

reversed text to rescore lattices that are just rescored with the forward LMs.

D. Data 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) Noise: We consider the impact of noise type, signal-to-noise ratio and reverberation. Especially, voice data reverberation is generated by simulated room impulse responses (RIRs) and speech convolution [20], of which the parameters include room size, room width, room height, speaker location, receiver location etc. We enlarge speech data by adding MUSAN noise with predefined SNR [21]. With the help of MUSAN dataset, we quadruple the amount of data using enhancement method consists of babble, music and noise. Additionally, RIRs dataset is expanded by reverberation enhancement a mentioned.

E. Pre-and-Post processing

1) 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. The SAD system follows work proposed in [22], described in detail in [23]. 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. For each audio input, firstly, we preprocess it and extract the fbank features. The subsystems output speech existence in every frame separately. Finally, the output passes the post-processing module and becomes the final output.

Additionally, we utilized an Subband Order Statistic Filters (OSFs) based VAD system [24].This system includes a noise reduction block that preceeds the VAD and uses OSFs to formulate a robust decision rule. Notice the distinct distributions in subband between speech and noise, this method significantly improves the performance in low signal-noise ratio conditions.

2) 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 bidirectionally [7].

3) System Fusion: After we obtain the recognition results for each system, we adopt two fusion methods to generate an output with reduced error rate. First, we adopt the lattice combination method [12], which uses minimum Bayes risk decoding to combine results under the same SAD system. Then, we try to merge the results obtained in the previous step with different SADs with the help of ROVER method [13]. Both methods are post-recognition processes which model 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.

4) 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.

F. Results on DEV

We test our systems with the 10h development set. The results are showed in Tab.II scored by Word Error Rate (WER). We show 4 single system results as well as their fusion results of each language and scoring criteria. The two acoustic model CNN-TDNN-F and CNN-TDNN-F-A are adopted and compared. Besides, we also list the WER of RNNLM rescored results and the augmented system results.

From Tab.II, we find that the augmented systems with MUSAN degrade to some extent while the systems with original data performs similarly. However, all the systems are useful to the fusion system.

IV. Constrained-plus Training Condition

The Constrained-plus Training condition follows the same training data restrictions as Constrained Training, but additionally allows publicly available and previously existing speech pre-trained models. The self-supervised learning frameworks, such as wav2vec2.0 [8], MockingJay [25], TERA [26], have obtained processing progress in speech recognition. Since there are 15 different languages in the challenge, a multilingual pre-trained model is a better choice than a monolingual one. We build our pipeline based on the wav2vec2.0 framework, which works well by fine-tuning the pre-trained model when the amount of labeled data is limited. The wav2vec2.0 model is composed of a multi-layer convolutional feature encoder, a context network which follows the Transformer architecture, and a quantization module to discretize the output of the

TABLE II
THE RESULTS (WER) OF DEV UNDER CONSTRAINED TRAINING CONDITION.

S1 and S2 employ the CNN-TDNN-F and CNN-TDNN-F-A respectively. S3 performs RNNLM rescoring based on the decoding results of S2. S4 is the augmented system by MUSAN with CNN-TDNN-F or CNN-TDNN-F-A. The lattice decoded from S1-S4 are fused by lattice combination, and the results are marked as $Fusion$.

Language	S1	S2	S3	S4	Fusion
Amharic_CIS	37.3	37.7	38.5	40.9	35.3
Cantonese_CIS	47.3	47.8	46.6	48.7	45.0
Farsi_CIS	54.4	54.2	55.0	58.0	52.6
Georgian_CIS	42.8	42.3	44.3	47.2	40.8
Guarani_CIS	41.0	41.8	41.6	44.9	38.9
Javanese_CIS	54.0	54.5	54.3	57.6	51.9
Kazakh_CIS	46.4	47.9	46.4	52.9	44.2
Kazakh_CSS	29.9	30.6	30.4	33.7	27.6
Kurmanji-kurdish_CIS	66.1	66.3	65.6	70.5	63.8
Mongolian_CIS	48.3	48.6	48.4	52.5	45.5
Pashto_CIS	47.0	47.2	46.3	51.2	44.1
Somali_CIS	55.9	56.6	56.4	59.7	53.8
Swahili_CIS	34.4	34.8	34.5	37.9	32.2
Swahili_CSS	27.5	28.0	29.8	29.3	26.4
Tagalog_CIS	42.2	42.9	41.9	44.1	40.0
Tagalog_CSS	33.4	33.3	30.9	34.1	28.5
Tamil_CIS	62.0	62.9	62.9	65.4	60.2
Vietnamese_CIS	47.4	47.0	46.4	50.3	44.2
Average	**45.4**	**45.8**	**45.6**	**48.8**	**43.1**

feature encoder to a finite set of speech representations via product quantization [8].

We use the open-source pre-trained model XLSR-53[5] as our baseline, which is a multilingual model trained with 56k hours audio data in 53 different languages from 3 datasets (Multilingual LibriSpeech [27], CommonVoice [28], Babel [10]).

A. Fine-tuning methods

For the basic workflow of a language, we fine-tune the pre-trained XLSR model with the labeled 10h data by adding a linear classifier on top of the model to optimize the Connectionist Temporal Classification (CTC) loss. The provided 10h development set is used to validate during fine-tuning. For the first 10k updates only the output classifier is trained, after which the Transformer is also updated. The feature encoder is not trained during fine-tuning.

We put character as the modeling unit for the languages with romanized spelling, such as Tagalog, Swahili and Javanese while the grapheme is utilized for the languages without romanized spelling, such as Tamil, Pashto and Amharic. The target tokens also include a word boundary token. For the case-insensitive scoring datasets, we transfer all uppercase letters into lowercase ones while keeping the letters unchanged for the case-sensitive scoring datasets. Besides, we deal with the transcripts text in two ways after removing all the speech aspects such as mispronunciations and non-speech aspects such as coughs. One is keeping all the words in the transcription of the speech while the other is filtering out the OOV words by the lexicon offered. According to our experiments, the former way performs better.

Since the pre-trained model is multilingual, it has the universality of multiple languages but lacks the speciality of a specific language. We further train the pre-trained model XLSR-53 with the target language unlabeled speech from the Build dataset in Babel program of the corresponding language being processed, though XLSR-53 has already employed the same part data during training for 8 in 12 Babel languages. At this phase, the whole model is optimized by contrastive loss augmented by a codebook diversity loss. Next, we fine-tune the obtained model with the labeled 10h data for the language. The 2-stage fine-tuning method is helpful to make the pre-trained model adapt to a single language efficiently.

B. Datasets

For the languages in Babel program, we use the speech data from the Build set of the corresponding language in an unsupervised way, which is described in Sec.IV-A. The detailed information about the speech data usage is showed in Tab.III.

TABLE III
THE EXTRA USED SPEECH DATA IN IARPA BABEL PROGRAM.

Language	Version	Duration
Amharic	IARPA-babel307b-v1.0b-build	43h
Cantonese	IARPA-babel101b-v0.4c-build	141h
Guarani	IARPA-babel305b-v1.0c-build	42h
Javanese	IARPA-babel402b-v1.0b-build	45h
Kazakh	IARPA-babel302b-v1.0a-build	39h
Kurmanji	IARPA-babel205b-v1.0a-build	41h
Mongolian	IARPA-babel401b-v2.0b-build	46h
Pashto	IARPA-babel104b-v0.bY-build	78h
Swahili	IARPA-babel202b-v1.0d-build	44h
Tagalog	IARPA-babel106-v0.2g-build	85h
Tamil	IARPA-babel204b-v1.1b-build	69h
Vietnamese	IARPA-babel107b-v0.7-build	88h

C. Experiment settings

When continuing the pre-training stage of XLSR-53, there is no layer drop. We optimize with Adam, warming up the learning rate for the first 32000 updates to a peak of 1×10^{-3} and then linearly decaying it. The maximum number of updates is 100k. The Gumbel softmax temperature is annealed from 2 to a minimum of 0.1 by a factor of 0.999995 at every update. The model is trained on a single GPU within 1.2M tokens a batch.

During fine-tuning, the learning rate is set to 1×10^{-3} as well. We optimize with Adam and a tri-state rate schedule where the learning rate is warmed up for the first 10% of updates, held constant for the next 40% and then linearly decayed for the remainder. We fine-tune on a single GPU with a batch of 1.28M samples.

D. Decoding, Alignment and Fusion

We perform SAD as well to infer efficiently and accurately. The two methods of SAD mentioned in Sec.III-E1 are adopted.

[5]https://dl.fbaipublicfiles.com/fairseq/wav2vec/xlsr_53_56k.pt

After fine-tuning, we decode with a 4-gram language model trained with the same text data in Sec.III-C. The LM weight is set to 1 with beam 500 for EVAL set and beam 5 for DEV set.

To obtain results with Conversation Time Mark (CTM) formats, we perform force-alignment with the GMM model trained before. Since the inference results are in one-best format, we set the confidence scores in CTM to 1.0 by default.

Finally, we fuse all the results from wav2vec2.0 framework as well as some results from systems in the Constrained Training Condition by ROVER. We find that results from different SAD methods are complementary to each other as the fusion performance improves a lot.

TABLE IV
THE RESULTS (WER) OF DEV DECODING WITH 4-GRAM LM OF BEAM 5 UNDER CONSTRAINED-PLUS TRAINING CONDITION.

FT is to fine-tune the XLSR model directly while $FT2$ continues training the XLSR model first, which is introduced as a 2-stage fine-tuning way above. FT_k means the texts used during fine-tuning keep all the words.

Language	**FT**	**FT2**	**FT_k**
Amharic_CIS	44.0	40.8	38.6
Guarani_CIS	46.2	42.5	41.3
Javanese_CIS	53.6	51.0	49.8
Kurmanji-kurdish_CIS	62.5	60.1	59.5
Mongolian_CIS	47.9	44.7	43.9
Pashto_CIS	45.2	41.8	37.9
Somali_CIS	53.9	-	56.3
Tamil_CIS	64.3	61.8	60.0
Vietnamese_CIS	40.2	35.9	36.6
Swahili_CIS	40.5	36.8	34.6
Swahili_CSS	48.6	45.4	38.1
Tagalog_CIS	45.0	41.5	39.5
Tagalog_CSS	43.6	42.1	34.3
Georgian_CIS	44.7	-	46.6
Kazakh_CIS	46.5	43.9	42.1
Kazakh_CSS	50.9	48.8	38.3
Farsi_CIS	46.3	-	47.2
Average-babel	**48.5**	**45.5**	**42.5**
Average-all	**48.5**	-	**43.8**

V. RESULTS

Our systems' performances of the 15 languages, 3 of which have extra case-sensitive scoring setting, under constrained training condition on the evaluation set are showed in Tab.V, which are released by NIST OpenASR scoring server. The submissions for constrained-plus training condition are displayed in Tab.VI. For each language, the submissions are composed of single systems and fusion systems which shows better performance.

VI. HARDWARE AND TIME DESCRIPTION

The basic hardware information for our systems is showed in Tab.VII. As for the required time, for each language under Constrained Training Condition, 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

TABLE V
THE THUEE SUBMISSIONS ON EVAL SET UNDER CONSTRAINED TRAINING CONDITION

The submission names omits the same prefix "2020_OPENASR21_evaluation_SYS-00949_THUEE"

Language	submission	WER
Amharic_CIS	20211108-101321-8653	0.452
	20211110-044759-3791	0.421
	20211111-065214-0660	**0.421**
Cantonese_CIS	20211108-020056-0415	0.453
	20211109-041913-9441	**0.449**
Farsi_CIS	20211109-020602-0166	0.822
	20211110-025256-5524	0.831
	20211110-212044-0942	0.828
	20211111-064847-0115	**0.814**
Georgian_CIS	20211109-021904-2667	0.482
	20211109-212733-7484	0.530
	20211110-024422-4784	**0.465**
Guarani_CIS	20211108-082805-1460	0.465
	20211109-214009-0433	0.490
	20211110-020249-1919	**0.442**
Javanese_CIS	20211108-081932-5133	0.535
	20211109-222017-8097	0.521
	20211110-015141-9145	0.521
	20211111-044515-7646	**0.520**
Kazakh_CIS	20211107-032129-6062	0.575
	20211108-103346-8794	0.578
	20211110-023608-1770	0.618
	20211110-031010-6373	0.548
	20211111-065245-0539	**0.545**
Kazakh_CSS	20211107-033318-3528	0.528
	20211109-223844-7195	**0.498**
Kurmanji-Kurdish_CIS	20211106-042417-1949	0.683
	20211108-213848-0585	**0.659**
	20211111-065333-1070	0.664
Mongolian_CIS	20211106-071158-5819	0.495
	20211107-013341-6114	0.532
	20211109-100710-9428	**0.453**
Pashto_CIS	20211106-072931-6612	0.491
	20211108-050918-4994	0.477
	20211111-014431-6892	**0.463**
Somali_CIS	20211106-033236-6055	0.618
	20211108-224115-0949	**0.580**
Swahili_CIS	20211108-091441-7648	0.380
	20211110-051655-7106	**0.352**
	20211110-210503-0470	0.379
	20211111-064741-1269	0.353
Swahili_CSS	20211108-092246-4903	0.464
	20211110-054839-9396	**0.437**
Tagalog_CIS	20211106-043355-7435	0.439
	20211108-035017-3710	0.423
	20211109-014953-5353	0.425
	20211109-225246-2840	**0.417**
Tagalog_CSS	20211106-220034-4920	0.518
	20211108-225632-7708	0.497
	220211110-010301-1583	**0.490**
Tamil_CIS	20211106-080915-0407	0.661
	20211108-234517-2131	0.645
	20211110-223148-0093	0.654
	20211111-065114-0877	**0.641**
Vietnamese_CIS	20211106-073607-7490	0.481
	20211109-013607-9726	0.453
	20211110-093507-3276	0.452
	20211111-065443-3885	**0.445**

TABLE VI
THE THUEE SUBMISSIONS ON EVAL SET UNDER CONSTRAINED-PLUS TRAINING CONDITION

The submission names omits the same prefix "OpenSAT-2020_OPENASR21_evaluation_SYS-00950_THUEE"

Language	submission	WER
Amharic_CIS	20211106-123024-2288	0.405
	20211109-194005-8482	0.398
	20211111-031217-0567	**0.373**
Farsi_CIS	20211106-104429-3292	0.639
	20211106-115828-2861	0.668
	20211111-030212-7077	**0.626**
Georgian_CIS	20211106-110246-9435	0.458
	20211106-113351-9522	0.456
	20211110-063036-8789	**0.404**
	20211110-065510-1662	0.408
Guarani_CIS	20211106-223450-2650	0.443
	20211109-044003-1945	0.424
	20211111-044346-2284	**0.406**
Javanese_CIS	20211106-001324-3009	0.512
	20211106-005520-2569	0.504
	20211109-084957-3105	**0.480**
Kazakh_CIS	20211106-114739-6624	0.488
	20211110-042800-2133	0.483
	20211110-044234-4422	**0.429**
Kazakh_CSS	20211106-102648-2444	0.533
	20211111-051458-8323	0.570
	20211111-053414-5171	**0.499**
Kurmanji-Kurdish_CIS	20211106-225210-2239	**0.615**
Mongolian_CIS	20211106-224314-6667	0.457
	20211109-091339-6520	0.431
	20211110-230022-8297	**0.411**
Pashto_CIS	20211107-001333-0470	0.440
	20211109-195924-5514	0.448
	20211109-224318-6051	**0.414**
Somali_CIS	20211107-000323-5918	0.584
	20211109-201205-7821	0.547
	20211111-062415-7045	**0.533**
Swahili_CIS	20211107-014317-0376	0.359
	20211110-025628-2966	0.333
	20211110-061439-1015	**0.315**
Swahili_CSS	20211106-230633-0618	**0.439**
	20211110-005616-8405	0.458
	20211111-060743-9406	0.446
Tagalog_CIS	20211106-112542-3907	0.422
	20211110-002419-8468	0.399
	20211111-015719-9038	**0.377**
Tagalog_CSS	20211106-100141-4759	**0.464**
	20211111-060255-3305	0.514
Tamil_CIS	20211107-005913-3307	0.616
	20211109-204114-6520	0.613
	20211111-064123-8424	**0.596**
Vietnamese_CIS	20211107-011444-8969	**0.373**

TABLE VII
HARDWARE DESCRIPTION

OS	Ubuntu 20.04.1 LTS 64-bit
CPU num	2
CPU description	40,Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz 80,Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz
GPU num	10
GPU description	GeForce RTX 3090 24GB
RAM per CPU	512GB
Disk storage	About 1TB

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. For the Constrained-plus Training Condition, the elapsed wall clock time is approximately 8 hours, which is mainly spent in fine-tuning pre-trained model with one GPU. The inference time is about 15 minutes with a single job for a language.

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.

[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] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," *CoRR*, vol. abs/1510.08484, 2015.

[7] 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.

[8] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2020, pp. 1–12.

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

[10] M. J. F. Gales, K. M. Knill, A. Ragni, and S. P. Rath, "Speech recognition and keyword spotting for low-resource languages: Babel project research at CUED," in *4th Workshop on Spoken Language Technologies for Under-resourced Languages, SLTU 2014, St. Petersburg, Russia, May 14-16, 2014*. ISCA, 2014, pp. 16–23.

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

[12] H. Xu, D. Povey, L. Mangu, and Z. Jie, "An improved consensus-like method for minimum bayes risk decoding and lattice combination," in *IEEE International Conference on Acoustics Speech Signal Processing*. IEEE, 2010, pp. 4938–4941.

[13] 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.

[14] M. Hasegawa-Johnson, L. Rolston, C. Goudeseune, G.-A. Levow, and K. Kirchhoff, "Grapheme-to-phoneme transduction for cross-language asr," in *International Conference on Statistical Language and Speech Processing*. Springer, 2020, pp. 3–19.

[15] AdolfVonKleist. (2021) Phonetisaurus. [Online]. Available: https://github.com/AdolfVonKleist/Phonetisaurus

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5220–5224, 2017.

[21] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *Computer Science*, 2015.

[22] 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.

[23] G.-X. Shi, W.-Q. Zhang, G.-B. Wang, J. Zhao, S.-Z. Chai, and Z.-Y. Zhao, "Timestamp-aligning and keyword-biasing end-to-end asr front-end for a kws system," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2021, no. 1, pp. 1–14, 2021.

[24] J. Ramírez, J. C. Segura, C. Benítez, A. De la Torre, and A. Rubio, "An effective subband osf-based vad with noise reduction for robust speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 6, pp. 1119–1129, 2005.

[25] A. T. Liu, S.-W. Yang, P.-H. Chi, P.-C. Hsu, and H.-Y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Barcelona, Spain, May 2020, pp. 6419–6423.

[26] A. T. Liu, S. Li, and H. Lee, "TERA: self-supervised learning of transformer encoder representation for speech," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 2351–2366, 2021.

[27] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "MLS: A largescale multilingual dataset for speech research," in *Interspeech 2020, 21th Annual Conference of the International Speech Communication Association, Shanghai, China, 25-29 October 2020*. ISCA, 2020, pp. 2757–2761.

[28] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds. European Language Resources Association, 2020, pp. 4218–4222.