The zxy System for OpenASR21 Challenge

Hongyu Song, Guolong Zhong, Ruoyu Wang, Chang Wang, Jun Du and Lirong Dai
National Engineering Laboratory for Speech and Language Information Processing (NEL-SLIP)
University of Science and Technology of China,
Hefei, Anhui, P. R. China
cndragon@mail.ustc.edu.cn

***Abstract*—This report describes the systems of 'zxy' (abbreviated from 'zui xing yun' which means 'the most fortunate' in mandarin) team in the IARPA Open Automatic Speech Recognition Challenge (OpenASR21). We participated in all fifteen languages in the constrained training condition, and seven languages in the unconstrained training condition. To increase the amount of training data in constrained condition, we adopted text-to-speech (TTS) techniques as an important data augmentation method which was shown to be very effective for low-resource languages. For the unconstrained condition, we trained several end-to-end (E2E) ASR models which had different encoder designs. During the evaluation stage, we focused more on case-sensitive scoring (CSS) tasks for particular languages. Based on the official ranking results at that time, we also submitted several results in constrained-plus condition based on confidence in our TTS based systems, though we did not use any pretrained models. Finally, our submitted system yielded good results on several tasks, including: swahili-constrained-css, tagalog-constrained-css and tagalog-constrained-plus.**

Keywords—low-resource languages, data augmentation, TTS, case-sensitive scoring, system fusion

I. INTRODUCTION

The goal of the OpenASR (Open Automatic Speech Recognition) Challenge is to assess the state of the art of ASR technologies for low-resource languages. Every language is separated into three training conditions: constrained, constrained-plus and unconstrained. We submitted results of Eval datasets for all languages in constrained condition and 7 languages in unconstrained condition. Case-insensitive scoring(CIS) Eval dataset will be offered for all fifteen languages and additional case-sensitive scoring (CSS) Eval dataset for three of the languages.

For the constrained condition, the hybrid model trained with Kaldi toolkit and the main architecture ResNet-TDNNF were used. The acoustic training data can only consist of the 10-hour Build datasets provided by the OpenASR21, so it's crucial that increasing the diversity of speech training data using unsupervised method and a small amount of speech data. The improvement of performance can be obviously observed after using some methods of data augmentation, such as utterance-level speed perturbation, training with text-to-speech TTS synthesized data, and features concatenating. We purchased a batch of publicly available text data from IARPA BABEL program [1], WILLTECH [2-7] and LDC (Linguistic Data Consortium) and crawled some public texts for different languages from the internet to optimize our language models. Finally we did decoding, rescoring, and fusing the recognition results obtained from different structures. By the way, we processed the languages for CSS with the same pipeline as CIS but there are some details different. For the unconstrained condition, we used extra thousands of speech data for pre-training, then we adopted the end-to-end (E2E) based model as the main strategy. Different designs of the encoder architecture were also explored. These models were fused and rescored. For both of the two conditions, the data processing methods were similar.

During the challenge, the main members of 'ustc_nelslip' team and 'zxy' team have overlap. The two teams had different priorities and plans in the evaluation stage: the 'ustc_nelslip' team focused more on constrained tasks, while the 'zxy' team experimented with more methods and tasks. For the unconstrained condition, because those tasks were very time-consuming, the two teams worked together in both model training and evaluation stages.

II. CONSTRAINED SYSTEM OF CIS

A. Training Data

For the constrained condition, the speech dataset allowed for training is only a 10-hour subset of the Build dataset provided by NIST for each language. For the text data, we also used transcriptions from the IARPA BABEL language packages, which lack Somali and Farsi. We crawled public texts from website for most languages to further optimize the language model. The collected texts were filtered and then added to the training corpus.

The datasets we used from IARPA BABEL language packages are listed as follows.

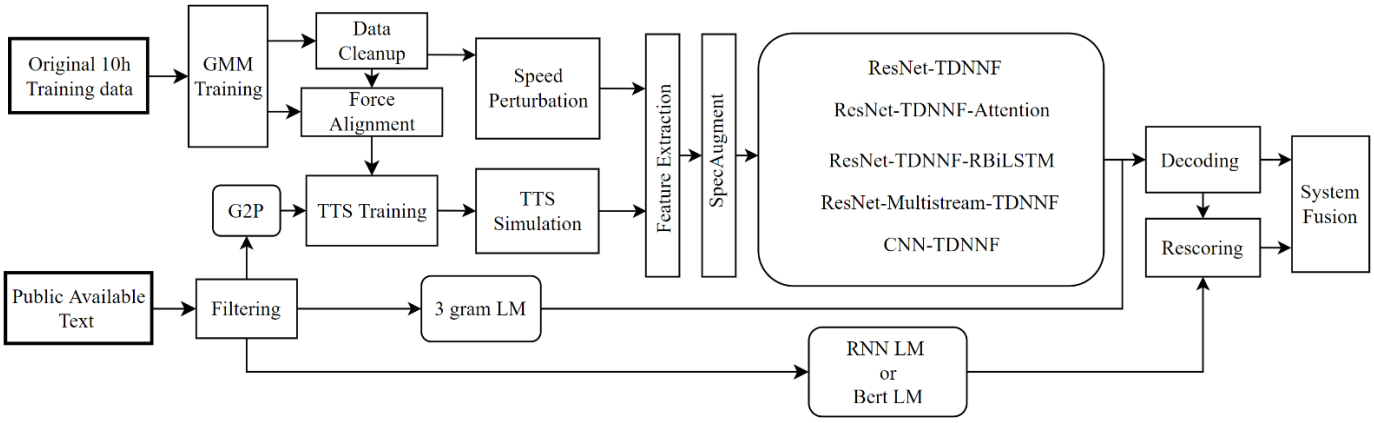

Fig. 1. Framework of the constrained system.

- Cantonese: IARPA-babel101b-v0.4c-build.
- Pashto: IARPA-babel104b-v0.bY-build.
- Tagalog: IARPA-babel106-v0.2g-build.
- Vietnamese: IARPA-babel107b-v0.7-build.
- Swahili: IARPA-babel202b-v1.0d-build.
- Tamil :IARPA-babel204b-v1.1b-build.
- Kurmanji-Kurdish: IARPA-babel205b-v1.0a-build.
- Kazakh: IARPA-babel302b-v1.0a-build.
- Guarani: IARPA-babel305b-v1.0c-build.
- Amharic: IARPA-babel307b-v1.0b-build.
- Mongolian: IARPA-babel401b-v2.0b-build.
- Javanese: IARPA-babel402b-v1.0b-build.
- Georgian: IARPA-babel404b-v1.0a-build.

B. *Overall System Diagram*

The overall framework of the constrained system is shown in Fig. 1. It contains several main parts including GMM modeling, data processing, acoustic model training, TTS system and language modeling. Details of each part will be described in following sections.

C. *Data Processing*

1) *Data cleaning*

The data cleaning was first applied following the recipe in Kaldi [8]. It aims to remove corrupted portions that are not accurate enough. The basic idea is to decode the training speech with an existing in-domain GMM acoustic model and build a biased language model from the reference transcripts, and then generate revised segmentation information [9].

2) *Speed and volume pertubation*

Speed and Volume perturbation [10] is as an effective data augmentation method for the raw data, which can alleviate overfitting and improve robustness of the models. We found that perturbating the speed with 3 factor (0.9,1.0,1.1) can give the best improvement. For volume perturbation, we used scale factors randomly chosen between 0.125 and 2.0.

3) *SpecAugment*

SpecAugment [11] is a simple data augmentation method for ASR which is applied directly to the input features of a neural network. We applied SpecAugment to the filterbank features, more details can be referred in [11].

D. *Acoustic Model Training*

1) *Training pipeline*

We built a hybrid DNN-HMM system using the Kaldi [8] toolkit. For the training data, all audio files in the training set are resampled to 8 kHz since most of the files are sampled at 8 kHz.

The rest 13 languages have corresponding IARPA BABEL language packs except for Somali and Farsi. Thus, the pronunciation lexicons were built based on BABEL. For Somali and Farsi, we used the lexicons provided by the 10-hour Build dataset.

A monophone GMM-HMM model was first trained with inputs of 13-dim mel-frequency cepstral coefficients (MFCCs) features (with 3 pitch features). Then, a context-dependent triphone model was trained, followed by Linear Discriminant Analysis (LDA) and Maximum Likelihood Linear Transform (MLLT) estimation. Finally, a speaker adaptive training (SAT) [12] model was trained with FMLLR [13]. We also estimated the probability of silence [14] from aligned training data during the training process.

For neural network training, alignments and numerator lattices were generated from the GMM-HMM model. We chose the ResNet-TDNNF network as our baseline acoustic model, which was trained using LF-MMI criterion [9] with cross-entropy (CE) regularization. Such pipeline is the so-called ‘chain model’ training in Kaldi.

2) *Baseline: ResNet-TDNNF model*

Our baseline acoustic model consists of stacks of Residual Network (ResNet) and Factorized Time Delay Neural Network (TDNNF). The Residual Network (ResNet) consists of convolutional layers, Batch Normalization (BN) and Rectified Linear Unit (ReLu) [15].

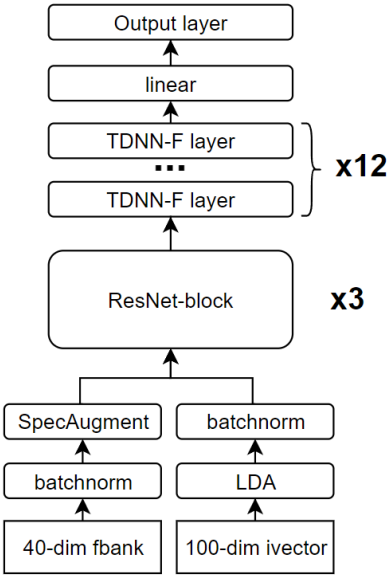

Fig. 2. ResNet-TDNNF architecture.

The popular TDNNF network, which is a fundamental component of our acoustic model. TDNNF is structurally similar to TDNN whose layers have been compressed via SVD, but are trained from a random start with one of the two factors of each matrix constrained to be semi-orthogonal in order to prevent instability in back-propagation [16]. A regular TDNNF block consists of a linear layer, an affine component, a ReLu nonlinear component, and a batch normalization operation followed by dropout.

Fig. 2 shows the architecture of ResNet-TDNNF. The network has two inputs: 40-dimensional Mel-filter bank coefficients (filter banks) features and 100-dimensional online i-vector features. In our systems, we trained a diagonal UBM-based i-vector extractor for speaker adaptation [17]. In order to adapt to the CNN structure, the 100-dim i-vectors are mapped to 200-dim by LDA transformation before being concatenated to the filterbanks.

We performed batch normalization on both i-vector and filterbank features. SpecAugment was used in training. Two inputs were transformed into spatial 40-dimensional planes (five for i-vector features, one for filterbank features) and combined with each other. The batch size was 128 or 64 with 6 epochs training in total. The initial learning rate was 0.001 and decayed during training, with the final learning rate 0.00005. In our experiments, we used the baseline acoustic model to evaluate different strategies and search for hyper-parameters.

3) Other architectures

In addition, we also trained other model architectures such as CNN-TDNNF, ResNet-Multistream-TDNNF [18], ResNet-TDNNF-Attention[19], and ResNet-TDNNF-RBiLSTM [19]. The details of them are listed as follows:

- CNN-TDNNF: 6-layer CNN blocks + 12-layer TDNNF
- ResNet-TDNNF-Attention: 7-layer ResNet + 12-layer TDNNF
- ResNet-TDNNF-BiLSTM: 7-layer ResNet + 3-layer TDNNF-RBiLSTM
- ResNet-Multistream-TDNNF: 7-layer ResNet + 12-layer TDNNF(x3)

These architectures were built using the LF-MMI training criterion by Kaldi. Lattice fusion [20] followed by Minimum Bayes Risk (MBR) decoding was performed to combine recognition results from different models using different architectures.

E. Increasing the Diversity of Training Data Acoustic

We found that the model trained on 10-hour data was easily overfitting and it was difficult to improve the performance of ASR by adjusting the parameters only. Therefore, we trained the system with various data augmentation methods to alleviate the overfitting problem. And the diversity of the system was of great benefit to the final fusion.

1) TTS synthesized data

In constrained condition, using Text-to-speech (TTS) outputs from TTS trained on designated constrained training data is allowed. We used Flow-TTS [21], which is a non-autoregressive end-to-end neural TTS model based on generative flow. Using a simple feed-forward network trained by jointly learning the alignment and spectrogram generation, Flow-TTS can achieve high-quality spectrogram generation.

We used a GMM model to get force alignment on the cleaned data, in order to get training resources for Flow-TTS training. The TTS model was used to synthesize MFCC (with pitch) and filterbank features separately. MFCC (with pitch) was used to generate the numerator lattices for chain model. And the filterbank features were used as input features to the chain model.

However, most words in the external texts were out-of-vocabulary (OOV). Therefore, we used the training lexicon provided by BABEL to train a grapheme-to-phoneme (G2P) model [22], which predicts the pronunciation of OOV words in external texts. Then the TTS model worked well with the predicted phonemes.

By using TTS synthesized data, we found that the performance of different structural models were improved. Table I shows the comparison of models trained without TTS Data and with TTS Data on the Pashto Dev set.

TABLE I. THE COMPARISON OF MODEL TRAINED WITHOUT TTS DATA AND WITH TTS DATA ON THE PASHTO DEV SET.

Model	WER (%)	
	Without TTS data	***With TTS data***
CNN-TDNNF	47.3	46.1
ResNet-TDNNF	47.1	45.6
ResNet-TDNNF-Attention	47.2	45.8
ResNet-Multistream-TDNNF	47.5	45.9
ResNet-TDNNF-RBiLSTM	50.0	47.3

2) Utterance-level speed and volume perturbation

The default perturbation process was performed for the whole original audio. However, the speaker's speed and volume usually change frequently in the dialogue scenarios. Therefore, we performed speed and volume perturbations at the utterance-level aiming to alleviate overfitting and improving the robustness of the model. To prevent over-perturbation, we calculated a series of allowable lengths based on the length of the utterance and the perturbation factors. When the length of perturbated utterances exceeds the allowed lengths, other suitable speed factors will be chosen to be used until the length meet the requirement.

Finally, we tripled the amount of training data. As for the utterance-level volume perturbation, we used a randomly chosen scale factor from 0.5 to 1.5.

3) Feature concatenating encoder representation

Considering that the end-to-end model and the hybrid DNN-HMM model are complementary in model fusion, as in [23]. We also trained the encoder-decoder (ED) model of the VGG-transformer [24]. To make the training of the ED model converge faster, we applied speed perturbation and Flow-TTS to increase the training data. We generated 12 times of the original training data with speed factors uniformly sampled from 0.8 to 1.2 at 0.25 intervals.

The 512-dim latent representations learned by the Encoder can be concatenated to 40-dim filterbank features, which brought further improvements to the final fusion system.

F. Languge Model

Language models trained on external texts are allowed to use in the OpenASR21 challenge. So we firstly use transcriptions from the "training" part of the IARPA BABEL program. For most languages, we also obtained large amounts of text data crawled from web. For convenience, the web-obtained data are denoted as ‘public’ data. However, these data are quite different from the data styles in BABEL. Data cleaning and filtering were performed on the public data.

Firstly, we performed cleaning on the public data. Chars that didn’t correspond to the language being processed were removed. Secondly, a domain classifier was trained to select data that have similar genres with BABEL from the public data [25]. For Cantonese, the domain classifier was migrated from the pre-training model Chinese BERT [26]. For the other languages, the domain classifiers were migrated from the multilingual pre-training model XLM-R [27]. Finally, we added colloquial noise to the extra text since the data genre in constrained condition is conversational telephone speech (CTS).

The first-pass language model was generated by interpolating an N-gram model trained on public data and another N-gram model trained on BABEL data. This process was done by using the SRILM [28]. As for language model (LM) rescoring, we adopted Transformer structure [29] for Cantonese and bidirectional RNN structure [30] for the rest languages.

For Cantonese, we continued to train Chinese BERT for several iterations using Cantonese public data and then fine-tuned using Cantonese BABEL data, aiming to transfer Chinese BERT to the domain of conversational style. We masked the whole word to make the model learn the inter-phrase relationship better. Compared to the first-pass decoding, the BERT-based [31] pretraining language model can bring absolute 0.3% WER improvement after rescoring on Cantonese.

For the rest languages, the models were initialized using the public data and fine-tuned using the BABEL training data corresponding to the language being processed. The RNNLM rescoring is effective in most language excepts Tamil, Vietnamese and Kurmanji-Kurdish. The RNNLM rescoring can bring at least absolute 0.2% WER improvement.

G. Voice Activity Detection (VAD)

Since the audios in CIS task are conversational telephone speech, most audios are without any environment noises. We firstly trained a TDNN-LSTM based VAD model for each language, using the implementations in Kaldi. However, the data-driven based model trained on only 10-hour data was not very stable according to our experiments. We found lots of strange missed error where the speech was very clear. Thus, we used an energy-based VAD method as a complement. The final detection result took the intersection of the two methods. Additionally, we expended the time regions in the front and back of each detected segment to prevent missing any speech. Each expanded duration is about 0.5 second.

Table II shows our VAD experiments on Cantonese and Pashto. The 'manual' represents using the time stamps provided in the transcripts of the Dev set. As we can see in the table, performance of VAD on Cantonese is similar to the manual time stamps. For Pashto, the VAD help to reduce the WER from 49.6% to 47.9%, which is mainly due to the reduction of false alarm errors in recognition results. In the evaluation stage, we directly migrated the strategies tuned on the Dev set for all languages.

H. Decoding

We used the WFST-based method for decoding in Kaldi. For the first-pass decoding, we used N-gram language model. The decoding beam was set to 16, while the beam used in lattice generation was 8.5. The LM weight was chosen from 7 to 17.

I. System Fusion

We used Lattice fusion followed by MBR decoding [20] to combine the recognition results of acoustic models trained with different architectures as well as different data augmentations. Benefiting from the effective compensation of different systems, the system fusion can greatly enhance the performance.

TABLE II. THE PERFORMANCE OF VAD FOR CANTONESE AND PASHTO ON DEV SET

Language	**WER (%)**	
	Manual	***VAD***
Cantonese	46.4	46.4
Pashto	49.6	47.9

J. Experiment Result

TABLE III. THE RESULTS OF EVAL SET

Language	WER (%)	
	Constrained	***Constrained-plus***
Amharic	42.5652	40.4499
Cantonese	48.3032	38.7994
Guarani	45.4791	43.1233
Javanese	51.0991	48.3802
Kurmanji-Kurdish	65.3694	61.8481
Mongolian	44.8922	41.8978
Pashto	47.3383	44.0087
Somali	58.5455	55.2239
Tamil	65.7738	63.2312
Vietnamese	43.4393	40.4380
Swahili	34.7295	32.8487
Tagalog	43.6949	41.3216
Georgian	42.3085	39.7809
Kazakh	53.3439	49.9072
Farsi	69.8557	69.7951

The final results of Eval set were generated by fusing systems of different architectures trained with TTS synthesized data, utterance-level speed and volume perturbation and features concatenating encoder representation.

Table III shows the final fusion results, which were released by the OpenASR21 scoring server. Note that we also submitted several results in constrained-plus condition, but we didn't use any pretrained models. Actually, the results of constrained-plus condition were obtained under constrained condition.

III. UNCONSTRAINED SYSTEM OF CIS

In the unconstrained condition, we are allowed to use additional speech and text training data from any language that can be publicly accessed. Because of the large amount of data, we used the encoder-decoder (ED) based end-to-end models as our unconstrained systems. In OpenASR2021, we only participated in Cantonese, Kazakh, Mongolian, Pashto, Tamil, Javanese, and Farsi.

We will describe our system in these several sections: 1) Data Pre-processing, 2) Modeling Unit, 3) Pretraining, 4) Model Training, 5) Language Model Rescoring, 6) Voice Activity Detection, 7) Force Alignment, and 8) Final Results.

A. Data Pre-processing

We used some external data which were purchased from many corporations. In addition, we conducted a series of data crawling to collect more speech-text data. Public videos which have subtitles can be seen as the target of data crawling, then pairs of audio and text are extracted. Due to time constraints, we only did data crawling for speech in Cantonese.

For Cantonese, the final training dataset consists of three main types: 140 hours data provided in IARPA BABEL package, 1,000 hours data from HUITING Tech Inc [32] and 3,000 hours crawled data. For the other six languages, their training data contain two main sources: from IARPA BABEL, and purchased from WILLTECH (Kazakh: 362 hours, Mongolian: 454 hours, Pashto: 315 hours, Tamil: 244 hours, Javanese: 332 hours, and Farsi: 324 hours) [2-7]. Dev sets for all languages kept the same with original OpenASR2021 datasets. In BABEL dataset, the sampling rate is 8 KHz but most of additional speech data are sampled at 16 KHz. As a result, we uniformly set the sampling rate to 16 KHz.

We carried out a series of operations to further enhance the diversity of original training data. Firstly, we used a method called as *long-term audio splicing*. For example, the non-speech segments were took as the separator to extract separate speech segments in BABEL, then different speech segments were stitched together. Considering the efficiency about data loading and model training, we set the maximum length of audio to be no more than 20 seconds. Secondly, we applied speed perturbation to both the original and long-term spliced data with the speed factors of 0.8, 1.0 and 1.2. Thirdly, we conducted noise augmentation on all audios. Finally, the overall data of every language reached more than 1,000 hours.

More importantly, we used the Flow-TTS based method to generate more acoustic data for training. To achieve it, we first sampled 100 hours data from original speech data to train TTS models. Then, we collected a large amount of publicly available text from websites. Using them, we can easily synthesize large amounts of filterbank features. During training, we randomly mixed real acoustic features and TTS synthesized features.

B. Modeling Unit

It's necessary to do data cleaning on the collected text data, aiming to determine proper modeling units. Firstly, all abnormal characters were removed. As a result, Kazakh had 43 characters, Mongolian had 38 characters, Pashto had 50 characters, Tamil had 50 characters, Javanese had 28 characters, and Farsi had 38 characters. Secondly, we tokenized the remaining texts using byte-pair-encoding (BPE) [33] and obtained 8,000 BPEs for each language except Cantonese. For Cantonese, we directly used Chinese characters which were more than 3,000 kinds and 26 English alphabets. Finally, all modeling units were sorted by frequency of occurrence.

C. Pretraining

A good pretrained model can be transferred quickly to other tasks. To accelerate our experiments in all seven languages, we performed pretraining works using two main types of datasets.

The first dataset is derived from IARPA BABEL corpora, including 25 languages. According to challenge rules, the overlapping parts between BABEL and OpenASR2021 were removed in advance. After speed perturbation and noise augmentation, approximately 8,000 hours of data were obtained. We directly used single characters in 25 languages as the

modeling units instead of using BPE. And all the characters in words were segmented with tag '<sep>'.

The second dataset consists mainly of Chinese and English, which includes many publicly available corpora such as Aishell [34], Aishell2 [35], Librispeech [36], TIMIT [37] and Switchboard [38]. The modeling units include 8,000 Chinese characters and 6,000 English BPEs.

For convenience, the model trained on the first dataset is denoted as 'Multi-lingual pretrained', and the other is 'Ch-En pretrained'. When using them, we only used the encoder of the pretrained models.

D. Model Training

We trained five different encoder-decoder based models. The optimization process follows a multi-task approach, one task is the final cross-entropy loss of the ED model, and the other one is the CTC loss of the encoder.

The Adam optimizer and warmup strategy were used and the initial learning rate was set to 0.0007. We also applied the SpecAugment and Scheduled Sampling [39] to make the system more robust. The E2E systems were trained using the open-source toolkit Fairseq [40]. Table IV describes the training setups about all five models which differs in the encoder design. As to the decoder part, five models use the same structure which contains six transformer layers. For each language, we trained all five types of models.

In the inference stage, we first conducted parameter averaging to each model and got five final models. When decoding, the beam size was set to 15. The posterior probabilities of all models were weighted and averaged. Meanwhile, those probabilities were also divided by the temperature constant for smoothing. The strategy was first validated in some languages and then extended to others. Table V shows the results of Dev sets on both Mongolian and Farsi.

Comparing Model 1 and Model 2 in Table IV and V, it's observed that training with TTS synthesized data can improve the overall performance. From the results of Model 2 and Model 5, the Multi-lingual pretrained model performs better than the Ch-En pretrained model, which can be attributed to more language coverage in training. Among all models, Model 5 yields the lowest WER. As the number of models increases in fusion stage, the recognition results consistently improve. The performance trend of the other five languages is similar to Table V.

E. Voice Activity Detection (VAD)

The VAD strategy in the unconstrained condition is similar to the constrained condition. Since an ED ASR model itself can also serve as a VAD module to some extent, we found that a simple energy-based VAD with proper thresholds could yield good and stable performance in unconstrained tasks. In the Dev set, replacing manual segments with energy-based VAD strategy could yield similar WERs on an ED ASR model. Finally, we migrated the same methods to the Eval set.

TABLE IV. THE TRAINING SETUPS OF DIFFERENT ED MODELS

Model	**Encoder**	**Training Data Type**	**Pretrained Model**
Model 1	9-layer DenseNet + 12-layer conformer block	Realistic	Ch-En
Model 2	9-layer DenseNet + 12-layer conformer block	Realistic + TTS	Ch-En
Model 3	4-layer Vggblock + 12 conformer layers	Realistic + TTS	Multi-lingual
Model 4	4-layer Vggblock + 12 transformer layers	Realistic + TTS	Multi-lingual
Model 5	9-layer DenseNet + 12 conformer layers	Realistic + TTS	Multi-lingual

TABLE V. THE RESULTS OF DIFFERENT MODELS AND MODEL FUSION

Dev Set	**WER (%)**	
	Mongolian	***Farsi***
Model 1	35.6	41.1
Model 2	33.6	39.5
Model 3	33.8	38.5
Model 4	34.2	40.9
Model 5	33.2	37.5
Model 1 + Model 2	32.5	37.5
Model 1 + Model 2 + Model3	30.8	35.2
Model 1 + Model 2 + Model3 + Model 4	29.5	34.8
Model 1 + Model 2 + Model 3 + Model 4 + Model 5	28.4	33.7

F. Language Model Rescoring

Since the one-best sequence in decoder output may not be the optimal result, we used an additional language model to do rescoring. We trained language models based on BERT. The training data consist of BABEL (training set) and the collected text data described above. The LM rescoring is useful in a few languages.

G. Force Alignment

As mentioned in the data pre-processing section, we used a large amount of long-term audios in training. During testing, it's better to conduct long-term splicing on VAD segments for matching such conditions. Segments spaced less than 1.5 seconds were sliced together, and the maximum length was set to 20 seconds. In our experiments, long-term testing can consistently get better results.

The final evaluation requires that the format of system output files is CTM. But the output sequences of an ED based system lacked fine-grained time information for every recognized word. To solve this problem, we also trained a hybrid DNN-HMM system using Kaldi, of which the model architecture is the same as the baseline used in constrained tasks. This hybrid system was only used to do force alignment on the recognized sequences generated by the ED system. As a result, we got the duration of each word and made final CTM files.

H. Final Result

TABLE VI. THE RESULTS OF EVAL SET

Language	WER (%)
Cantonese	30.1925
Kazakh	38.1197
Mongolian	31.8812
Pashto	34.0277
Tamil	56.9548
Javanese	44.4218
Farsi	52.0046

The performances of our final fusion systems of Eval set are presented in Table VI.

IV. CONSTRAINED SYSTEM OF CSS

Case-sensitive scoring tasks are offered about Kazakh, Swahili, and Tagalog in OpenASR21, which means words capitalized differently between the hypothesis with the reference will not count as a match.

For CSS, all datasets stem from the IARPA MATERIAL program. MATERIAL datasets are separated into three data genres, conversational telephone speech (CTS), news broadcast (NB) and topical broadcast (TB). Table VII show the compositions of these two kinds of datasets.

CTS data in BABEL are from conversations between two persons over the telephone on a topic of their choosing. Conversations vary in length, up to approximately 10 minutes. CTS data in MATERIAL are either from a subset of BABEL CTS data or newer sets collected and annotated using the BABEL methodology.

MATERIAL NB data contain audio segments of approximately 2.5 minutes from widely distributed broadcasts as well as regional and local news covering news topics and current affairs. The broadcasts are recorded with studio quality, and the speech context could be formal or informal depending on the segments.

MATERIAL TB data are similar to NB data in terms of audio quality and speech characteristics, but more of in-depth topics. Each recording is approximately five minutes. Table VIII show the durations of the different data genres in these three languages.

For CSS task, we adopted the same strategies of data processing, acoustic model training, VAD, decoding, and system fusion as that in the CIS task.

A. Training Data

For each language, the 10-hour Build set provided by OpenASR21 would be used for training the acoustic model. We also selected the training transcriptions of BABEL as additional text data to train the language model (see Table IX).

TABLE VII. THE COMPOSITONS OF CIS AND CSS DATASETS

Dataset	Resources	Data Genre
CIS	BABEL (MATERIAL for Somali and Farsi)	CTS
CSS	BABEL, MATERIAL	CTS, NB, TB

TABLE VIII. TOTAL DURAITON ABOUT THE TRAINING DATA IN CSS LANGUAGES .

Language	Data Genre	Duration
Kazakh	NB + TB	5 hours
	CTS	5 hours
Swahili	NB + TB	5.5 hours
	CTS	4.5 hours
Tagalog	NB + TB	3 hours
	CTS	7 hours

TABLE IX. IARPA BABEL LANGUAGE PACKS USED FOR ADDITIONAL TEXT DATA

Language	LDC ID	Language Package
Kazakh	LDC2018S13	IARPA-babel302b-v1.0a-build
Swahili	LDC2017S05	IARPA-babel202b-v1.0d-build
Tagalog	LDC2016S13	IARPA-babel106-v0.2g-build

B. Modeling

According to the OpenASR21_Evaluation_Plan [41], words capitalized differently from the reference transcriptions will not count as a match, so we kept the cases of words in corpus, lexicons and recognition results, which means the final words we submitted would consist of upper and lower cases. In order to obtain more available words, for each language, we combined the corresponding lexicons from the Build dataset of OpenASR21 and the IARPA BABEL language packs as the final lexicon we used.

We tented to treat the prefix of file names as speaker-ids. For CSS data, the NB/TB audios are named differently from the CTS audios. The CTS audios were processed the same way in CIS, while the NB/TB data were processed in two ways: one is mapping all the NB/TB audios to the same speaker; another is mapping each NB/TB audio to its own speaker. For Kazakh, the first operation reduced WER of Dev set by absolute 2.8% compared to the second operation, but brought no improvement to the other two languages.

C. Language Model

We found that part of the Build datasets of OpenASR21 is the subset of the training datasets of BABEL. So we removed the extra repeated transcriptions to avoid the overfitting of language model caused by the overlapping data. Then the rest transcriptions were adopted as the corpus for language modeling.

The language model was an N-gram model trained on the corpus using the SRILM [28]. We knew the BABEL datasets

only consist of CTS, but it really reduced the WER of Dev set compared to the language model which only used the text in 10-hour build corpus, about absolute 0.8% for Kazakh and Swahili, 2.4% for Tagalog.

We also attempted the same language modeling method used in CIS to build the second-pass language models without adjustment because of the poor time. The public data made no contribution to the WER because of the unreasonable data composition, which needs to be further tuned and researched.

D. Confidence Filtering

We applied confidence filtering to the ASR results obtained from lattice. Words are filtered according to their confidence scores. The threshold was set to be 0.2, which means the recognition results with confidence scores below the threshold would be discarded. If the deletion errors is high, we will tend to set a small threshold. The higher the threshold, the less substitution and insertion errors. We found that confidence filtering bring about at least 0.1% improvement in WER for each language.

E. Experiment Results

When testing on Dev set, for example, we got different results for 'Manila'. The results are shown in Fig. 3. For the lower case, the substitute error was generated. But for the upper case, the recognized word was correct. So it is necessary to get the right form of each recognized word for the case-sensitive scoring.

Table X shows our final fusion results on Eval datasets.

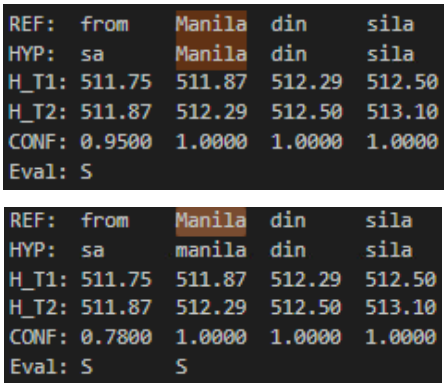

Fig. 3. Evaluation results of 'Manila'.

TABLE X. THE WER ON EVAL DATASETS

Language	WER(%)		
	Constrained	***Constrained-plus***	***Unconstrained***
Swahili-CSS	43.4985	43.9941	——
Tagalog-CSS	46.177	46.2981	——
Kazakh-CSS	54.6797	52.8451	——

HARDWARE AND TIME REQUIREMENT

TABLE XI. THE HARDWARE DESCRIPTION OF A SINGLE SERVER

OS	CentOS 7.2 64-bit
CPU num	48
CPU description	Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
GPU num	4
GPU description	Tesla V100-PCIE 32GB
RAM	128 GB
Disk storage	10 TB

The hardware description of a single server is shown in Table XI. In constrained condition, we performed all training experiments on a single server. For each language, the elapsed wall-clock time is approximately 20 hours for a single whole system. It takes 40 minutes for GMM training, 5 hours for TTS model training on 1 GPU and 4 hours for chain model training on 1 GPU. With TTS synthesized data, it takes 6 hours for GMM training, 7 hours for chain model training on 4 GPU in parallel. GPU resources were only used for NN acoustic model training. Running decoding pipeline using GPU on the Eval set takes about 30 minutes. And the maximum memory consumption in decoding was around 12 GB.

In unconstrained condition, the processing time of pretraining on 25 languages using 24 GPUs is about 40 hours. For each language, it takes 10 hours for TTS training on 4 GPUs, 40 hours for a single ED model training on 12 GPUs. The total processing time required is about 80 hours.

REFERENCES

[1] (2011) Babel program. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/babel

[2] https://www.futve.com/#/recommend/details/?id=738&classId=20

[3] https://www.futve.com/#/recommend/details/?id=732&classId=20

[4] https://www.futve.com/#/recommend/details/?id=742&classId=20

[5] https://www.futve.com/#/recommend/details/?id=735&classId=20

[6] https://www.futve.com/#/recommend/details/?id=744&classId=20

[7] https://www.futve.com/#/recommend/details/?id=746&classId=20

[8] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al., "The kaldi speech recognition toolkit," in IEEE 2011 workshop on automatic speech recognition and understanding, no. CONF.IEEE Signal Processing Society, 2011.

[9] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, and S. Wang, Y.and Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in Proc. of INTERSPEECH, 2016, pp. 2751–2755.

[10] Ko T., Peddinti V., Povey D., & Khudanpur S., "Audio augmentation for speech recognition," in Proc. INTERSPEECH, Dresden, Germany, Sep. 2015, pp. 3586–3589.

[11] D. S. Park, W. Chan, Y. Zhang, Y. Chiu, C. C. Zoph, B. Cubuk et al., "SpecAugment: A simple data augmentation method for automatic speech recognition," in Proc. INTERSPEECH, Graz, Austria, Sep. 2019, pp. 2613–2617.

[12] V. V. Digilakis, D. Rtischev and L. G. Neumeyer, “Speaker adaptation using constrained estimation of Gaussian mixtures”, in IEEETransactions on Speech and Audio Processing, vol. 3, pp. 357-366, 1995

[13] M.J.F Gales, “Maximum likelihood linear transformations for HMM-based speech recognition,” Computer Speech and Language, vol. 12, pp. 75-98, 1998.

[14] "Pronunciation and Silence Probability Modeling for ASR", Guoguo Chen, Hainan Xu, Minhua Wu, Daniel Povey and Sanjeev Khudanpur, Interspeech 2015

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[16] Povey, Daniel & Cheng, Gaofeng & Wang, Yiming & Li, Ke & Xu, Hainan & Yarmohammadi, Mahsa & Khudanpur, Sanjeev. (2018). Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. 3743-3747. 10.21437/Interspeech.2018-1417.

[17] 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.

[18] K. J. Han, J. Pan, V. K. N. Tadala, T. Ma and D. Povey, "Multistream CNN for Robust Acoustic Modeling," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 6873-6877, doi: 10.1109/ICASSP39728.2021.9414639.

[19] Chai, Li & Du, Jun & Liu, Di-Yuan & Yanhui, tu & Lee, Chin-Hui. (2021). Acoustic Modeling for Multi-Array Conversational Speech Recognition in the Chime-6 Challenge. 912-918. 10.1109/SLT48900.2021.9383628.

[20] H. Xu, D. Povey, L. Mangu, and J. Zhu, “Minimum bayes risk decoding and system combination based on a recursion for edit distance[J]. Computer Speech & Language, 2011, 25(4):802-828.

[21] C. Miao, S. Liang, M. Chen, J. Ma, S. Wang and J. Xiao, "Flow-TTS: A Non-Autoregressive Network for Text to Speech Based on Flow," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 7209-7213, doi: 10.1109/ICASSP40776.2020.9054484.

[22] M. Bisani and H. Ney: "Joint-Sequence Models for Grapheme-to-Phoneme Conversion". Speech Communication, Volume 50, Issue 5, May 2008, Pages 434-451

[23] Liu, Andy & Li, Shang-Wen & Lee, Hung-yi. (2020). TERA: Self-Supervised Learning of Transformer Encoder Representation for Speech.

[24] Mohamed, Abdelrahman & Okhonko, Dmytro & Zettlemoyer, Luke. (2019). Transformers with convolutional context for ASR.

[25] Zhao, J., Lv, Z., Han, A., Wang, G.-B., Shi, G., Kang, J., Yan, J., Hu, P., Huang, S., Zhang, W.-Q. (2021) The TNT Team System Descriptions of Cantonese and Mongolian for IARPA OpenASR20. Proc. Interspeech 2021, 4344-4348, doi: 10.21437/Interspeech.2021-1063

[26] https://github.com/ymcui/Chinese-BERT-wwm

[27] https://github.com/facebookresearch/XLM

[28] A. Stolcke, “SRILM - an extensible language modeling toolkit,” in proc. ICSLP - interspeech, Denver, Colorado, USA, Sep. 2002.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[30] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, “A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5929–5933.

[31] Kenton, Jacob Devlin Ming-Wei Chang, and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT. 2019.

[32] http://www.huitingtech.com/en/dataInfo.action?id=1005

[33] Sennrich R , Haddow B , Birch A . Neural Machine Translation of Rare Words with Subword Units[J]. Computer Science, 2015.

[34] Bu H, Du J, Na X, et al. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline[C] 2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA). IEEE, 2017: 1-5.

[35] Du J , Na X , Liu X , et al. AISHELL-2: Transforming Mandarin ASR Research Into Industrial Scale[J]. 2018.

[36] Panayotov V, Chen G, Povey D, et al. Librispeech: an asr corpus based on public domain audio books[C] 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2015: 5206-5210.

[37] Victor Z , Seneff S , Glass J . TIMIT Acoustic-phonetic Continuous Speech Corpus[C] Linguistic Data Consortium. 1993.

[38] Godfrey, J. J. , E. C. Holliman , and J. Mcdaniel . "SWITCHBOARD: telephone speech corpus for research and development." Acoustics, Speech, and Signal Processing, 1992. ICASSP-92. 1992 IEEE International Conference on IEEE, 2002.

[39] Bengio, Samy, et al. "Scheduled sampling for sequence prediction with recurrent Neural networks." Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1. 2015.

[40] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, Michael Auli: fairseq: A Fast, Extensible Toolkit for Sequence Modeling. NAACL-HLT (Demonstrations) 2019: 48-53

[41] (2021) OpenASR21_Evaluation_Plan. [Online]. Available: https://www.nist.gov/document/openasr21-challenge-evaluation-plan