# The USTC_NELSLIP System for OpenASR21 Challenge

Guolong Zhong, Hongyu Song, 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

Lei Sun, Diyuan Liu, Genshun Wan, Kai Han, Haitao Tang, Li Chai, Li Yan, Jingxuan Zhang, Minghui Wu, Dan Liu, Jia Pan, Xin Fang, Junhua Liu and Jianqing Gao
*Research & Development Group*
*IFLYTEK CO.LTD.*
Hefei, Anhui, P. R. China
leisun8@iflytek.com

***Abstract*—This paper describes our system participating in the IARPA Open Automatic Speech Recognition Challenge (OpenASR21). Our submissions cover both the constrained condition (all fifteen languages) and unconstrained condition (seven languages). For the constrained condition, we adopt the hybrid ASR system for acoustic modeling. Importantly, we train text-to-speech (TTS) models in order to increase the size of training data. As for unconstrained condition, we choose the end-to-end (E2E) ASR architecture which can benefit more from adequate training data than hybrid ASR systems. We also train several candidate models which have different encoders. Finally, model fusion is used to further improve the overall performance. According to the results on the evaluation sets, there are still a lot of room for improvement in the field of low-resource ASR.**

***Keywords—low-resource languages, end-to-end ASR, data augmentation, TTS, system fusion***

## I. INTRODUCTION

Deep learning based automatic speech recognition (ASR) system requires a large amount of annotated data to work reasonably well, but most of the languages in the world have very limited training data. The goal of the OpenASR (Open Automatic Speech Recognition) Challenge, organized by NIST, is to assess the state of the art of ASR technologies for low-resource languages [1].

The challenge offers three different training conditions: constrained, constrained-plus, and unconstrained. In the evaluation stage, we participated in all the 15 languages in the constrained condition and 7 languages in the unconstrained condition. All of our submissions were scored case-insensitively.

For the constrained condition, the training dataset in each language contains only 10-hour speech. Additional text data from publicly available resources are permissible during training. For language model (LM), we use additional text data from IARPA BABEL [2] and other public text collected from websites. Our system was built based on hybrid DNN-HMM approach using Kaldi toolkit. We used ResNet-TDNNF as our baseline acoustic model which was used to explore different strategies. In order to increase the amount and diversity of training data, we explored some conventional data augmentation methods such as speed perturbation, volume perturbation, and Spec-Augment. Specially, we trained text-to-speech (TTS) models for all languages to generate more acoustic features for training. Our experiments suggest that TTS data can effectively improve the performance of low-resource ASR system. Based on the diversity of training data and model structures, we built different systems from different combinations among them. LM rescoring was also applied after the first pass decoding. Using lattice fusion, recognition results of various models were combined.

For the unconstrained condition, participants are allowed to use additional publicly available speech and text data from any languages. In our system, we relied primarily on the IARPA BABEL corpora and conducted data augmentation based on them. Additionally, we collected many other data for each language. Unlike using the hybrid ASR system in the constrained condition, end-to-end (E2E) based models were adopted as the main strategy. Different designs of the encoder architecture were also explored. LM rescoring was applied to the final fusion systems.

Finally, we participated all fifteen language in the constrained condition and seven languages in the unconstrained condition. In this paper, we will share our experience about the system details.

## II. CONSTRAINED SYSTEM

In this section, we will describe main components of our submitted systems under constrained condition.

### *A. Training Data*

For the constrained condition, the speech dataset which is permissible for training is only a 10-hour subset of the Build dataset provided by NIST for each language. For text data, we also use transcriptions from IARPA BABEL language packs which lack Somali and Farsi (see Table I). To further optimize the language model, we crawled public texts of most languages from websites. The crawled texts were filtered and then added to the training corpus.

TABLE I. IARPA BABEL LANGUAGE PACKS USED FOR ADDITIONAL TEXT TRAINING DATA

| Language | LDC ID | Language Package |
|---|---|---|
| Cantonese | LDC2016S02 | IARPA-babel101b-v0.4c-build |
| Pashto | LDC2016S09 | IARPA-babel104b-v0.bY-build |
| Tagalog | LDC2016S13 | IARPA-babel106-v0.2g-build |
| Vietnamese | LDC2017S01 | IARPA-babel107b-v0.7-build |
| Swahili | LDC2017S05 | IARPA-babel202b-v1.0d-build |

| Language | LDC ID | Language Package |
|---|---|---|
| Tamil | LDC2017S13 | IARPA-babel204b-v1.1b-build |
| Kurmanji-Kurdish | LDC2017S22 | IARPA-babel205b-v1.0a-build |
| Kazakh | LDC2018S13 | IARPA-babel302b-v1.0a-build |
| Guarani | LDC2019S08 | IARPA-babel305b-v1.0c-build |
| Amharic | LDC2019S22 | IARPA-babel307b-v1.0b-build |
| Mongolian | LDC2020S10 | IARPA-babel401b-v2.0b-build |
| Javanese | LDC2020S07 | IARPA-babel402b-v1.0b-build |
| Georgian | LDC2016S12 | 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 such as data processing, GMM modeling, chain-model training, TTS system, language modeling. Details of each part will be described in the following sections.

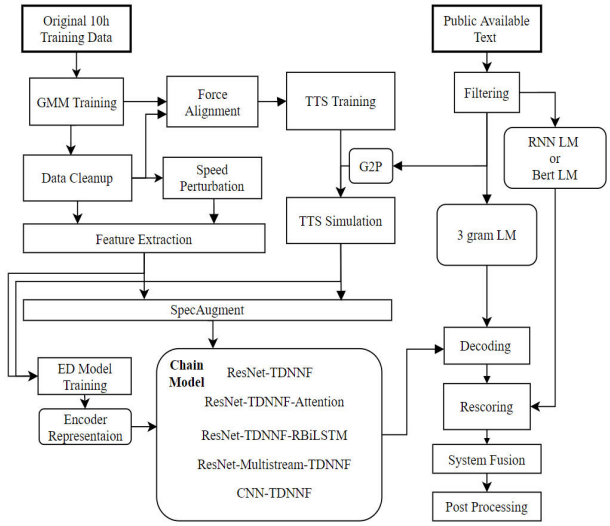


Fig. 1. Framework of the Constrained System.

## C. Data Processing

The dataset consists of conversations between two persons and the audios are distributed for each channel separately. The length of each audio is about 10 minutes so we segmented the training data according to the time stamps provided in the training transcripts.

### 1) Data cleaning

Data cleaning was first applied which followed frequently-used recipes in Kaldi [3]. 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 a biased language model built from the reference transcripts, and then generate the revised segmentation information [4].

### 2) Speed and volume perturbation

Speed and volume perturbations [5] are effective data augmentation methods for low-resource tasks, which can alleviate overfitting problems and improve the model robustness. We found that perturbating the speed of speech with 3 factors (i.e., 0.9,1.0,1.1) achieved the best performance. For volume perturbation, scale factors were randomly selected from 0.125 to 2.0.

### 3) SpecAugment

SpecAugment [6] is a simple data augmentation method for ASR which is applied directly to the feature inputs of a neural network. We applied SpecAugment to the filterbank features, more details can be referred to [6].

## D. Acoustic Model Training

### 1) Training procedures

For low-resource condition, building a pure E2E ASR system is not an appropriate choice since the overfitting problem lies in anywhere. Thus, we chose the classic hybrid DNN-HMM based structure which was built using Kaldi [3] toolkit. Before training, all the audio files in the training sets were resampled to 8 kHz.

In OpenASR21, most languages have corresponding IARPA BABEL language packs, except for Somali and Farsi. Thus, most pronunciation lexicons were directly built based on BABEL. For Somali and Farsi, we built the pronunciation lexicons using the lexicons provided by the 10-hour Build dataset provided in the challenge. Special non-speech tags in the reference transcripts were clustered into four categories (<v-noise>, <noise>, <silence>, <oov>). Additionally, characters '-' were removed.

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) [7] model was trained with FMLLR [8]. We also estimated the probability of silence [9] 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 [4] 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) [10].

With the presence of residual connections, the ResNet can improve the convergence in training and tackle the vanishing gradient problem so that we can train a deeper network.

The popular TDNNF network is another basic component of our acoustic model which has achieved great success in hybrid systems [11]. 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 [12]. Thus, it can greatly reduce the number of model parameters. A regular TDNN-F block consists of a linear layer, an affine component, an ReLU nonlinearity component, and batch normalization operation followed by dropout.

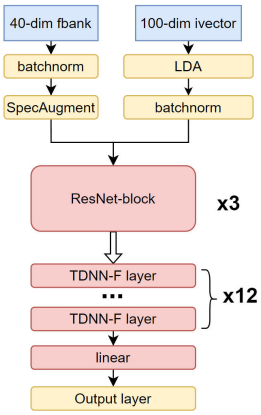


Fig. 2. The ResNet-TDNNF architecture.

Fig.2 shows the architecture of ResNet-TDNNF. The network has two inputs: 40-dimensional Mel-filter bank coefficients (filter bank) features and 100-dimensional online i-vector features. In our systems, we trained an i-vector extractor based on a diagonal UBM for speaker adaptation [13]. When dumping i-vectors for training, we split speakers up into multiple copies by limiting the number of utterances per speaker to be 5 at most. In order to adapt to the CNN structure, the 100-dim i-vectors are mapped to 200-dim by LDA transformation before concatenating with filterbank features.

We applied batch normalization to 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. **Note that** the results are only comparable with each other within the same table.

*3) Other architectures*

Besides, we also trained other model architectures such as CNN-TDNNF, ResNet-Multistream-TDNNF [14], ResNet-TDNNF-Attention [15], and ResNet-TDNNF-RBiLSTM [15]. Their details are listed as follows:

- CNN-TDNNF: 6-layer CNN blocks + 12-layer TDNNF
- ResNet-TDNNF-Attention: 7-layer ResBlock + 12-layer TDNNF + 1-layer Self-attention
- ResNet-TDNNF-RBiLSTM: 7-layer ResBlock + 3-layer TDNNF-RBiLSTM
- ResNet-Multistream-TDNNF: 7-layer ResBlock + 12-layer TDNNF (x3)

They were all built with LF-MMI training criterion using the Kaldi toolkit. Lattice fusion [16] 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*

Though using traditional data augmentation methods described in Section II.C, we found that the model was still easy to overfit. The main reason can be attributed to the fact that only 10 hours of original training data are available.

In this part, we make several attempts by training models with different types of acoustic data. Moreover, the diversity of systems is of great benefit to the final fusion.

*1) TTS synthesized data*

In our system, Flow-TTS [17], a non-autoregressive E2E neural TTS model based on generative flow, was adopted. Flow-TTS can achieve high-quality spectrogram generation by using a simple feed-forward network, which is trained by learning the relationship between the alignments and spectrograms. To prepare the essential materials needed in TTS training, we needed to perform force alignment on the cleaned data using the GMM model. In the inference stage, the TTS model was used to synthesize acoustic features which correspond to the collected text.

There are still some out-of-vocabulary (OOV) words in the external text. Therefore, we used the training lexicon provided by BABEL for training a grapheme-to-phoneme (G2P) model [18] which can predict pronunciations for OOV words. But most of the time, we just used sentences where all the words are in the provided dictionary. During inference, we converted the text into phoneme sequences.

According to our experiments, larger amounts of TTS data do not necessarily guarantee better performance of the ASR model. The ratio of real data to TTS data is crucial. We performed a series of experiments with different data ratios, and the results indicated the optimal ratio was about 1:1.5.

Table II shows the results on the Cantonese development set. By utilizing synthesized data, we found the performance of different models was consistently improved. Also, it's observed that adding TTS data yields more performance improvements for models which have a wider receptive field, such as ResNet-Multistream-TDNNF, from 47.1% to 45.4%.

TABLE II. THE RESULTS OF DIFFERENT MODELS TRAINED WITHOUT TTS DATA AND WITH TTS DATA ON THE CANTONESE DEV SET

| **Model** | **WER (%)** | |
|---|---|---|
| | ***Without TTS data*** | ***With TTS data*** |
| CNN-TDNNF | 47.3 | 45.6 |
| ResNet-TDNNF | 46.9 | 45.3 |
| ResNet-Multistream-TDNNF | 47.1 | 45.4 |

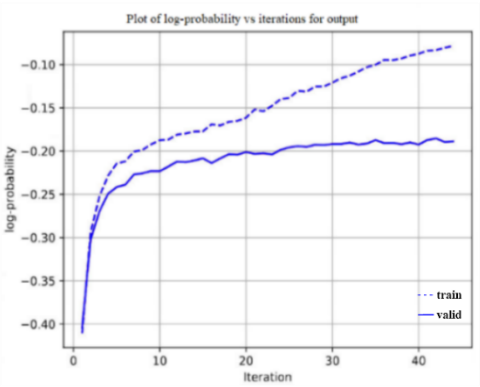


Fig. 3. The log-probability vs iterations of without-TTS model.

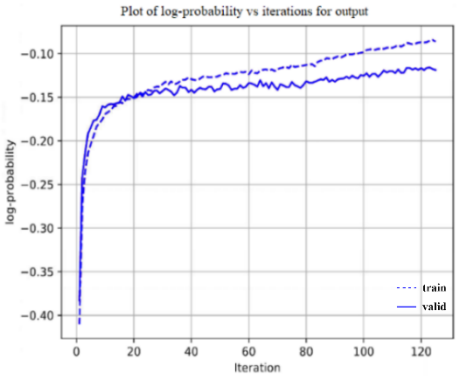


Fig. 4. The log-probability vs iterations of with-TTS model.

Fig.3 and Fig. 4 presents log-probabilities during training. The dashed line represents the log-probability of training diagnostic set and the solid line represents the log-probability of validation subset. Comparing Fig. 3 with Fig. 4, it can be seen that the overfitting problem of model training was alleviated.

*2) Adaptive speed perturbation*

The default process of perturbations was conducted on one whole raw audio. However, the speed and volume of the speaker's speech will change frequently in the conversational scenario. Therefore, we performed speed and volume perturbation at the utterance level. We perturbed speeds of utterances to force their lengths to some allowed lengths spaced by a factor of 1.15. We calculated a series of allowed lengths based on the length of utterances and the perturbation factors. Utterances that were too short (or too long) were kept unchanged. The quantity of training data was also increased by 3 times.

TABLE III. THE COMPARISION OF DIFFERENT SPEED PERTURBATION METHODS ON THE CANTONESE DEV SET

| Method | WER (%) |
|---|---|
| Baseline (3x speed perturbation) | 46.9 |
| Adaptive-speed | 46.3 |
| Adaptive-speed-volumn | 46.1 |

This method is called as *adaptive speed perturbation.* As for utterance-level volume perturbation, we used scale factors randomly selected from 0.5 to 1.5. The system trained in the way of adaptive speed perturbation is better than that trained by simply doing 3x speed perturbation, as shown in Table III. However, this procedure was time-consuming so we only applied it to a unique model for final model fusion.

*3) Transformer based Encoder representation*

As suggested in [19], the features extracted from the encoders have shown great representation ability in several tasks. Hence, we trained an encoder-decoder (ED) model based on the VGG-transformer [20]. To satisfy the great data demand in training an E2E ASR model, we applied speed perturbation and Flow-TTS to augment training data. We produced 12 times of the original training data with speed factors uniformly sampled from 0.8 to 1.2 at 0.25 intervals. SpecAugment and dropout were used for training. The encoder was used as a feature extractor. Finally, 512-dim latent representations learned in the encoder can be concatenated with the 40-dim filterbanks as the fused feature. Models trained with such fused features were seen as a unique category, which was prepared for final model fusion.

## *F. Language Model*

We firstly used transcriptions from the "training" part of the IARPA BABEL program. We also obtained a large amount of publicly available text data collected from the web for most languages. However, these data are quite different from BABEL data styles. So we performed careful data cleaning and filtering on the web-obtained data.

Taking Cantonese as an example, we first performed data cleaning. Chars that were not belonging to the language being processed were removed. Sentences that were repeated, or containing more than 70% of repeated words, 70% of numbers as well as 20% of out-of-vocabulary words, were removed with a probability of 0.8. Secondly, a domain classifier was trained for selecting data that have similar genres with BABEL from the public data. The positive samples were BABEL data and the negative ones were randomly selected from the public data. These two types of samples had equal amount when combining them as the training dataset [21]. For convenience, the selected web-obtained data are denoted as 'public' data.

For Cantonese, the domain classifier was fine-tuned from the pre-training model Chinese-BERT-wwm [22]. For the other languages, the domain classifiers were adapted from the multilingual pre-trained model XLM-R [23].

Finally, we added colloquial noise to the public text since the data genre in the constrained condition is conversational telephone speech. Specifically, modal particles were inserted at the beginning, the middle and the end of the sentence with a probability of 0.2, 0.1 and 0.2; reduplicated words (no more than 3) were inserted with a probability of 0.1; and low-information words with small inverse document frequency (IDF) were removed with a probability of 0.1.

We did some experiments with the baseline acoustic model. Table IV shows the PPL of the N-gram language models trained with different types of data. We can see that a reasonable data processing strategy has a large impact on the model effect.

TABLE IV. THE PPL OF THE N-GRAM LANGUAGE MODELS TRAINED WITH DIFFERENT TYPES OF DATA

| Training Data | PPL |
|---|---|
| BABEL | 108 |
| L1: public cleaned | 657 |
| L2: L1 + domain classifier | 545 |
| L3: L1 + domain classifier + add noise | 353 |
| BABEL + L3 N-gram LM interpolation | 102 |

TABLE V. THE NUM OF UTTERANCES OF THE PUBLIC DATA

| Training Data | Number of Utterances (k) |
|---|---|
| Cantonese | 10,000 |
| Kazakh | 6,000 |
| Pashto | 4,000 |
| Javanese | 180 |
| Mongolian | 1,670 |
| Farsi | 4,000 |
| Amharic | 260 |
| Georgian | 630 |
| Somali | 30 |
| Swahili | 60 |
| Tagalog | 30 |
| Tamil | 620 |

Table V shows the number of utterances of the public data after data cleaning and filtering.

For the first pass decoding, the language model was generated by interpolating an N-gram model trained on public data and another N-gram model trained on BABEL data. This was conducted using the SRILM [24]. Due to the large gap in style between public data and BABEL data, the weight of the BABEL language model was set to 0.8 to ensure the model effect. The interpolated language model was applied to Cantonese, Mongolian, Farsi, and Kazakh, and shows no improvement on other languages. Table VI shows the performance of the first pass interpolated language model for Cantonese, Mongolian, Farsi, and Kazakh.

TABLE VI. THE PERFORMANCE OF THE FIRST PASS INTERPOLATED LANGUAGE MODEL FOR CANTONESE, MONGOLIAN, FARSI, AND KAZAKH

| Language | WER (%) | | |
|---|---|---|---|
| | ***BABEL N-gram LM*** | ***BABEL + Public N-gram LM*** | ***Improvement*** |
| Cantonese | 46.7 | 46.3 | 0.4 |
| Kazakh | 52.0 | 51.7 | 0.3 |
| Mongolian | 58.1 | 57.7 | 0.4 |
| Farsi | 58.3 | 57.6 | 0.7 |

For LM rescoring, we adopted Transformer model [25] for Cantonese and bidirectional RNN model [26] for the rest languages.

For Cantonese, Chinese-BERT-wwm was continued to be trained for several iterations using Cantonese public data and then fine-tuned using Cantonese BABEL data. Then the model was transferred 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 pretrained language model brought absolute 0.5% WER improvement after rescoring on Cantonese as table VII shows.

For other 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 was effective in most language, except Tamil, Vietnamese and Kurmanji-Kurdish. However, the Transformer structure has no advantage over bidirectional RNN structure for all languages. Table VIII shows the performance of RNNLM rescoring in the rest language.

In addition, we found that the WER is much higher than CER because our word segmentation was inconsistent with that in reference transcripts. Taking Cantonese as an example, the reference text is "系 唔系 边度 睇 下", while the hypothesis is "系唔系 边 度 睇下". So, we used all BABEL data to train a word segmentation model based on the BERT pretrained model [27], and then re-segment the recognized words, making it much closer to the reference transcripts. The word segmentation model can also be applied to the result in the unconstrained condition. After word segmentation, about an absolute 0.3% WER reduction can be achieved on the development set while the CER remains unchanged. However, due to the time limit, we didn't apply the word segmentation model to the evaluation set.

TABLE VII. THE PERFORMANCE OF RESCORING WITH BERT LANGUAGE MODEL FOR CANTONESE ON DEV SET

| Language | WER (%) | | |
|---|---|---|---|
| | ***Baseline*** | ***+ Bert Rescore*** | ***Improvement*** |
| Cantonese | 46.9 | 46.4 | 0.5 |

TABLE VIII. THE PERFORMANCE OF RESCORING WITH BI-RNN LANGUAGE MODEL ON DEV SET

| Language | WER (%) | | |
|---|---|---|---|
| | ***Baseline*** | ***+ Bi-RNN Rescore*** | ***Improvement*** |
| Kazakh | 52 | 51.6 | 0.4 |
| Pashto | 47.9 | 46.9 | 1.0 |
| Javanese | 61.4 | 60.9 | 0.5 |
| Mongolian | 58.1 | 57.8 | 0.3 |
| Farsi | 58.3 | 56.9 | 1.4 |
| Amharic | 37.2 | 36.7 | 0.5 |
| Georgian | 39.5 | 39.0 | 0.5 |
| Somali | 57.2 | 57.0 | 0.2 |

| Language | WER (%) | | |
|---|---|---|---|
| | ***Baseline*** | ***+ Bi-RNN Rescore*** | ***Improvement*** |
| Swahili | 33.5 | 32.8 | 0.7 |
| Tagalog | 45.1 | 44.0 | 1.1 |
| Guarani | 41.4 | 40.0 | 1.4 |

TABLE IX. 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 |

### *G. Voice Activity Detection (VAD)*

Since the audios in OpenASR21 are recorded from telephone conversations, most recordings are without any environmental noises. We first trained a TDNN-LSTM [28] based VAD model for each language, using the implementations in Kaldi. However, according to our experiments, the data-driven based model trained on the limited 10-hour data is not very stable for some recordings. We found lots of strange missed errors where the speech was very clear. Thus, we lowered the requirements for the VAD module and used an energy-based VAD method as a complement. The final detection result takes the combination of the two methods. Additionally, we extended the time regions in the front and back of each detected segment to prevent missing any speech. Each expanded duration is about 0.5 seconds.

Table IX shows our VAD experiments on Cantonese and Pashto. The 'manual' represents using the time stamps provided in the transcripts of the development set. The manual time stamps only segment the entire audio without detecting speech regions, so false alarm error in recognition results is easy to appear. As we can see in the table, the performance of VAD on Cantonese is similar to manual time stamps. For Pashto, the VAD helps 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 development set for all languages.

### *H. Decoding*

In our systems, we used a 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. We found that language model weightings (lmwt) and word insertion penalties (wip) had great impacts on recognition results. We set lmwt to be 11 and wip to be 0 by experience.

### *I. System Fusion*

We used Lattice fusion followed by MBR decoding [16] to combine the recognition results of acoustic models trained with different architectures as well as different types of data. Table X shows the fusion results of different systems and their respective results after the first pass decoding for Cantonese on the Dev set.

TABLE X. THE PERFORMANCE OF DIFFERENT SYSTEMS FOR CANTONESE ON DEV SET

| Model | WER (%) | | |
|---|---|---|---|
| | ***Without TTS data*** | ***With TTS data*** | ***Concatenated Encoder representations*** |
| CNN-TDNNF | 45.9 | 43.5 | -- |
| ResNet-TDNNF | 45.1 | 42.6 | 46.8 |
| ResNet-TDNNF-Attention | 46.7 | 45.6 | 46.3 |
| ResNet-Multistream-TDNNF | 45.9 | 42.8 | -- |
| ResNet-TDNNF-RBiLSTM | 46.8 | 44.2 | 48.1 |
| ResNet-TDNNF (Adaptive-speed-volumn) | 46.1 | -- | -- |
| Fusion | 39.6 | | |

Due to the time limit, we didn't train systems with all kinds of data augmentation methods. We fused all the systems in table X because we found such a fusion strategy can bring the best result. Note that the results in table X are directly obtained by the first pass decoding (with VAD, interpolated language model and hyper-parameters adjustment), but without LM rescoring or post processing which we have applied to the evaluation set.

We can see that system fusion can greatly enhance the performance, benefiting from the effective compensation of different systems.

### *J. Post Processing*

#### *1) Confidence filtering*

We applied confidence filtering to the ASR results obtained from lattice. Words were filtered according to their confidence scores [27]. The threshold was set within the range from 0 to 0.5, which means the recognition results with confidence scores below the threshold would be discarded. If the deletion errors were high, a small threshold would be used. The higher the threshold was, the fewer substitution and insertion errors the recognition result had. We found that confidence filtering with appropriate threshold brought about at least an absolute 0.2% improvement in WER for most languages and an absolute 0.5% improvement for some languages such as Kurmanji-Kurdish, Tamil, Amharic.

#### *2) Removing underline*

The lexicon provided in the references materials contains some words with underlines, and we modeled the whole words, resulting in the recognition results also with underlines. Then, we split the words with underlines in the recognition results into multiple words. For example, 'I_B_M' was split into 'I B M', which brought improvement in Javanese, Kurmanji-Kurdish and Swahili.

### *K. Final Results*

In the evaluation period, the lattices of the first pass decoding were rescored and then fused with the rescored lattices. We found that the fusion result of rescored lattices is not always better than the fusion result of the first pass decoding, but combining both two fusion results can further improve the

performance. The final system output was generated by fusing nearly 15 systems and post processing. Note that the fusion strategies were tailored for each language during the evaluation.

TABLE XI. THE RESULTS ON BOTH DEV AND EVAL SET

| Language | WER (%) | |
|---|---|---|
| | ***Dev*** | ***Eval*** |
| Amharic | 32.0 | 39.9 |
| Cantonese | 38.2 | 37.6 |
| Guarani | 36.4 | 42.6 |
| Javanese | 47.8 | 48.1 |
| Kurmanji-Kurdish | 61.4 | 61.7 |
| Mongolian | 41.3 | 41.0 |
| Pashto | 41.4 | 43.2 |
| Somali | 52.7 | 55.6 |
| Tamil | 57.7 | 62.3 |
| Vietnamese | 40.9 | 40.3 |
| Swahili | 29.5 | 32.4 |
| Tagalog | 39.3 | 40.4 |
| Georgian | 34.9 | 39.2 |
| Kazakh | 40.1 | 50.0 |
| Farsi | 49.5 | 68.0 |

Table XI shows the final results of our fusion system. The results on Dev set were obtained by Sclite [29] with Reference File Format (STM) generated by ourselves. The results on Eval set were released by the OpenASR21 scoring server. As we can see, after LM rescoring and post processing, the lattice fusion achieved 1.4% absolute reduction of WER for Cantonese (from 39.6% to 38.2%).

## III. UNCONSTRAINED SYSTEM

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 based E2E models as our unconstrained systems. In OpenASR21, we only participated in Cantonese, Kazakh, Mongolian, Pashto, Tamil, Javanese, and Farsi.

We will describe our system in several sections, including: 1) Data Pre-processing, 2) Modeling Unit, 3) Pre-training, 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, publicly available data that were bought from many corporations. Besides, we also 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-hour data provided in IARPA BABEL package, 1,000-hour data from HUITING Tech Inc [30] and 3,000-hour crawled data. For the other six languages, their training data contain two main sources: from IARPA BABEL, and bought from WILLTECH [31-36] (Kazakh: 362 hours, Mongolian: 454 hours, Pashto: 315 hours, Tamil: 244 hours, Javanese: 332 hours, and Farsi: 324 hours). Development sets for all languages keep the same with original OpenASR21 datasets. In BABEL dataset, the sampling rate is 8 KHz. However, most of additional speech data are sampled at 16 KHz. As a result, we uniformly set the sampling rate to 16 KHz.

Based on the original audios, we took a series of operations to further enhance the diversity of training data. Firstly, we used a method called as *long-term audio splicing*. For example, the non-speech segments were taken as the separator to extract separate speech segments in BABEL, then different speech segments were stitched together. Considering the efficiency of 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 features.

### *B. Modeling Unit*

To determine proper modeling units, it's necessary to do data clean-up on the collected text data. 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. Then, we tokenized the remaining texts using byte-pair-encoding (BPE) [37] and obtained 8,000 BPEs for each language except Cantonese. For Cantonese, we directly used Chinese characters which are more than 3,000 kinds and 26 English letters. Finally, all modeling units were sorted according to frequency of occurrence.

### *C. Pre-training*

As we know, a good pretrained model can be transferred quickly to other tasks. To accelerate our experiments in all seven languages, we conducted the pre-training works using two main types of datasets.

The first dataset is derived from IARPA BABEL corpora, including 25 languages. The overlapping parts between BABEL and OpenASR21 dataset were removed in advance. After speed perturbation and noise augmentation, approximately 8,000 hours of data were obtained. Instead of using BPE, we directly used single characters in 25 languages as the modeling units. 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 [38], Aishell2 [39], Librispeech [40], TIMIT [41] and Switchboard [42]. 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*

During 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.

We used the Adam optimizer and warmup strategy. The initial learning rate was set to 0.0007. We also applied the SpecAugment and Scheduled Sampling [43] to make the system more robust. The E2E systems were trained using the open-source toolkit Fairseq [44]. Table XII describes the training setups about all five models which differ 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. In Table XIII, the results of development set on both Mongolian and Farsi are listed.

TABLE XII. 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 XIII. THE RESULTS OF DIFFERENT MODELS AND MODEL FUSION FOR MONGOLIAN AND FARSI

| **OpenASR21 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 |

As seen in Table XII and XIII, the comparison between Model 1 and Model 2 shows that adding TTS synthesized data in training 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 during fusion stage, the recognition results consistently improve. The performance trend of the other five languages is similar to Table XIII.

### *E. 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 two language models which are based on N-gram and BERT, respectively. The training data consist of BABEL (training set) and the collected text data described above. In Table XIV, results of development set of the two language models in multi-model fusion on both Mongolian and Farsi are listed.

It can be seen that the language model does not contribute much in terms of final WERs. However, we still used BERT for rescoring in the evaluation stage, especially for Cantonese where the recognized characters are in great need of reordering.

### *F. 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 development set, replacing manual segments with an energy-based VAD strategy could yield similar WERs on an ED ASR model. Finally, we migrated the same method to the evaluation set.

TABLE XIV. THE RESULTS OF DIFFERENT LANGUAGE MODELS IN MODEL FUSION FOR MONGOLIAN AND FARSI

| **OpenASR21 Dev Set** | **Language Model** | **WER (%)** | |
|---|---|---|---|
| | | ***Mongolian*** | ***Farsi*** |
| Model 1 + Model 2 + Model 3 + Model 4 + Model 5 | N-gram | 28.40 | 33.70 |
| | BERT | 28.39 | 33.69 |

### G. Force Alignment

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

However, the final evaluation requires that system results should follow the CTM format. Though an ED based system can give more precise output sequences than the traditional hybrid ASR system, it lacks fine-grained time information for every recognized word.

To solve this problem, for unconstrained tasks we also trained a hybrid DNN-HMM system using Kaldi. The model architecture is the same with our constrained works. This hybrid system was only used to do force alignment on the recognized sequences generated by the ED system. As the Fig. 5 shown, we got the duration of each word and made final CTM files. Another way to do force alignment is to take the usage of the CTC output. The non-empty characters output of all CTC branch from all five models were voted frame by frame. An optimal frame sequence was selected and aligned with the recognition results.

In the development set, the performance of hybrid-system based force alignment was stable, while the CTC-branch based method is unstable. Finally, we used CTC-branch based force alignment only on Kazakh and Tamil, and other languages use hybrid-system based results. In evaluation stage, we migrated the same strategies.

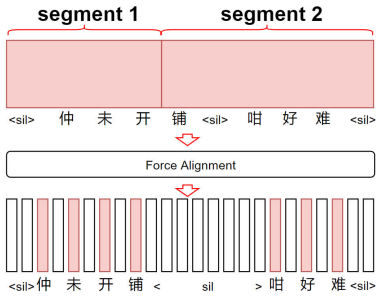


Fig. 5. Force alignment used on the recognized sequences of ED system.

### H. Final Result

The final results of both the development set and evaluation set are listed in Table XV. As we can see, the WER of the evaluation set is higher than that of the development set. Among them, Farsi is the most anomalous one, with a gap of nearly 20.

TABLE XV. THE RESULTS ON BOTH DEV AND EVAL SET

| Language | WER (%) | |
|---|---|---|
| | ***Dev*** | ***Eval*** |
| Cantonese | 22.0 | 26.6 |
| Kazakh | 29.5 | 37.4 |
| Mongolian | 28.4 | 31.5 |
| Pashto | 30.7 | 33.9 |
| Tamil | 51.2 | 55.9 |
| Javanese | 37.8 | 39.5 |
| Farsi | 33.7 | 52.0 |

## HARDWARE AND TIME REQUIREMENT

The hardware description of a single server is shown in Table XVI. 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 evaluation set takes about 30 minutes. And the maximum memory consumption in decoding was around 12 GB.

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

TABLE XVI. 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 |

## REFERENCES


[1] (2021) OpenASR21 Challenge. [Online]. Available: https://www.nist.gov/itl/iad/mig/openasr-challenge

[2] (2011) Babel program. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/babel

[3] 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.

[4] 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.

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

[6] 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.

[7] 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

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

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

[10] 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.

[11] G. Hinton, L. Deng, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, et al., "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," in IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, Nov. 2012, doi: 10.1109/MSP.2012.2205597.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

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

[21] 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

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

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

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

[25] 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.

[26] 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.

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

[28] Peddinti V, Wang Y, Povey D, et al. Low latency acoustic modeling using temporal convolution and LSTMs[J]. IEEE Signal Processing Letters, 2017, 25(3): 373-377.

[29] (2018) SCTK, the NIST scoring toolkit. [Online]. Available: https://github.com/usnistgov/SCTK

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

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

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

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

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

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

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

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

[38] 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.

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

[40] 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.

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

[42] 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.

[43] 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.

[44] 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