The CHiME-7 Challenge: System Description and Performance of NeMo Team's DASR System

Tae Jin Park, He Huang, Ante Jukić, Kunal Dhawan, Krishna C. Puvvada, Nithin Koluguri, Nikolay Karpov, Aleksandr Laptev, Jagadeesh Balam and Boris Ginsburg

NVIDIA, Santa Clara, USA

{taejinp,heh,ajukic,kdhawan,kpuvvada,nkoluguri,nkarpov,alaptev,jbalam,bginsburg}@nvidia.com

Abstract

We present the NVIDIA NeMo team's multi-channel speech recognition system for the 7th CHiME Challenge Distant Automatic Speech Recognition (DASR) Task, focusing on the development of a multi-channel, multi-speaker speech recognition system tailored to transcribe speech from distributed microphones and microphone arrays. The system predominantly comprises of the following integral modules: the Speaker Diarization Module, Multi-channel Audio Front-End Processing Module, and the ASR Module. These components collectively establish a cascading system, meticulously processing multi-channel and multi-speaker audio input. Moreover, this paper highlights the comprehensive optimization process that significantly enhanced our system's performance. Our team's submission is largely based on NeMo toolkits and will be publicly available.

1. Introduction

The landscape of conversational artificial intelligence (AI) has seen significant advancements in recent years, with an increased emphasis on applications such as automatic speech recognition (ASR), text-to-speech synthesis (TTS), and extensive use of large language models (LLMs). Among the numerous open-source toolkits in this domain, NVIDIA's NeMo is emerging as a pivotal toolkit for conversational AI, particularly ASR models. A hallmark of the NeMo toolkit is its ability to leverage prior work, in terms of both code and pretrained models, thereby enhancing the development and evaluation process more efficient and reproducible. Our submission for *CHiME-7* DASR task is largely based on the NeMo models and code bases which are originally built for single-channel tasks. Most importantly, we publicly release our submission based on NeMo toolkit so that anyone could test and improve the proposed system.

In the context of practical applications, this paper explores NVIDIA NeMo team in the 7th CHiME Challenge. Specifically, we shed light on the intricacies of their multi-channel speech recognition system, designed to transcribespeech picked up from multiple, distributed microphones and arrays. While our focus is honed in on Track 1, encompassing distant automatic speech recognition (DASR), the details of this system are revealed through its principal components, such as the speaker diarization module, the multi-channel (MC) Audio front-end processing module, and the ASR Module. Collectively, these modules create a cascade that adeptly handles multi-channel, multi-speaker audio input. We will also highlight the optimization methods employed to boost the system's efficiency, as showcased on the development set.

This paper is constructed in the following structure. In Section 2 we explain the details of each signal processing chain and

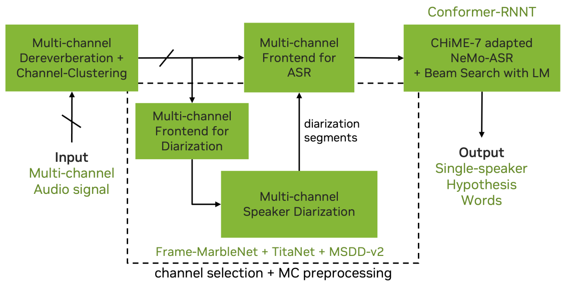
Multi-channel Dereverberation + Channel-Clustering
Multi-channel Frontend for ASR
Conformer-RNNT
CHiME-7 adapted NeMo-ASR + Beam Search with LM
Multi-channel Frontend for Diarization
diarization segments
Input
Multi-channel Audio signal
Output
Single-speaker Hypothesis Words
Multi-channel Speaker Diarization
Frame-MarbleNet + TitaNet + MSDD-v2
channel selection + MC preprocessing

Figure 1: *Dataflow of the NeMo Team's CHiME-7 Submission.*

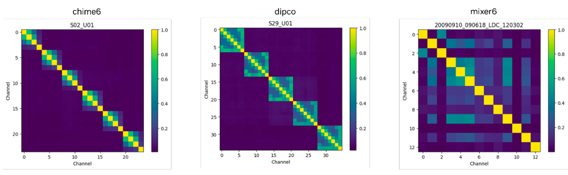

Figure 2: *Channel clustering: $\bar{\Gamma}$ computed on a single session from CHiME-6 (left), DiPCo (middle), and Mixer 6 (right) development subsets.*

neural models that comprise our DASR system. In Section 3 we discuss the hyper-parameter optimization method used to adjust the non-differentiable parameters affecting the overall performance. In Section 4, we present the evaluation results on the dev and eval set of the challenge dataset. Finally, we conclude and discuss the future work in Section 5.

2. Proposed DASR system

The Fig. 6 illustrates the data flow of our submission for the DASR task. In the highest level, our system consists of the following modules: dereverberation with channel clustering, front-end for diarization, speaker diarization module, front-end for ASR, ASR decoder module.

2.1. Dereverberation with channel-clustering

This module aims to perform dereverberation and channel clustering on the raw multi-channel audio input. The goal of the dereverberation module is to reduce the impact of room reverberation on diarization performance, similar to previous submissions [1]. The objective of the channel clustering module is to decrease the number of audio streams utilized for multi-channel diarization. The multi-channel audio signals are processed using the weighted prediction error-based dereverberation for multiple-input multiple-output (MIMO) as presented in [2] and implemented in NeMo toolkit [3]. Block-wise MIMO dereverberation is conducted over 40-second windows with a two-second window overlap. This process utilizes STFT with 64 ms window length, 75% overlap, 10 frame filters, a 3-frame prediction delay and 10 iterations. Using the processed multi-

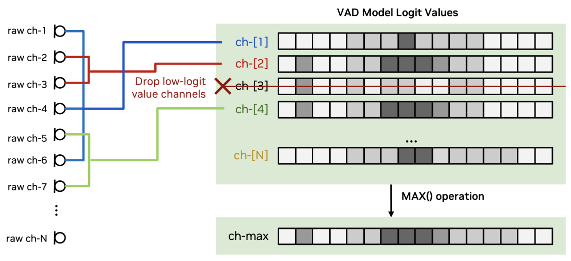

Figure 3: *Multi-channel VAD is applied on clustered channels while low-logit channels are dropped. Finally, frame-wise max pooling is applied across the remaining clustered channels.*

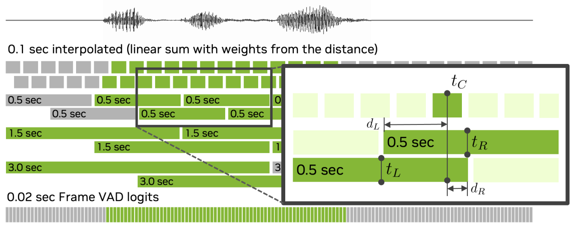
0.1 sec interpolated (linear sum with weights from the distance)
0.5 sec
1.5 sec
3.0 sec
0.02 sec Frame VAD logits
t_C
d_L
t_R
t_L
d_R

Figure 4: *Speech signal and its corresponding multi-scale segments for speaker embeddings. The frame-VAD logits will mask the only frames detected as speech.*

channel signal, we compute a magnitude-squared coherence (MSC) matrix, $\mathbf{\Gamma}(f) \in \mathbb{R}^{M \times M}$, where M represents the number of channels. This approach is consistent with the methodology in [4], and the elements in $\mathbf{\Gamma}(f)$ are:

$$\{\mathbf{\Gamma}(f)\}_{ij} = \frac{|S_{ij}(f)|^2}{S_{ii}(f)S_{jj}(f)}, \tag{1}$$

where $S_{ij}(f)$ is the cross-power spectral density between channels i and j. The average MSC matrix $\bar{\mathbf{\Gamma}}$ is obtained by averaging over frequency subbands f between 300 Hz and 3.5 kHz. Fig. 2 depicts examples of $\bar{\mathbf{\Gamma}}$ for a single session from each of the subsets of the development set. It can be observed that the patterns in $\bar{\mathbf{\Gamma}}$ correspond to different microphone array configurations used for recording the data for each of the subsets. Clustering of the channels is obtained by applying maximum eigengap spectral clustering (NME-SC) [5] on $\bar{\mathbf{\Gamma}}$. Note that the number of clusters is estimated automatically using normalized maximum eigengap [5], and channel clustering is performed for each session independently. The obtained channel clusters are used to reduce the number of audio streams for multi-channel diarization. Signals within each cluster are averaged, and these output streams are used as the input for the following steps of multi-channel speaker diarization.

2.2. Speaker diarization

2.2.1. Multi-channel Voice Activity Detection

We employ a convolution-based voice activity detection (VAD) model to predict speech probability for each 20 ms block of the input audio signal [6]. The model is randomly initialized and trained on a combination of the CHiME-6 [7] training subset and an additional simulated dataset. For validation, we use dataset comprises the CHiME-6 development subset as well as 50 h of simulated audio data. The simulated data is generated using the NeMo multi-speaker data simulator [8] on VoxCeleb1&2 datasets [9, 10].This results in a total of 2,000 hours of audio, of which approximately 30% is silence. Additionally, the model training incorporates SpecAugment [11] and noise augmentation through MUSAN [12]. During inference,

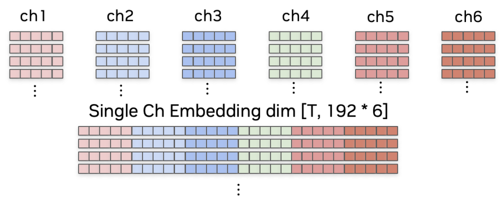

Figure 5: *Multi-channel speaker embedding vectors are concatenated to form an multi-channel super-vector that incorporates speaker traits from all channels.*

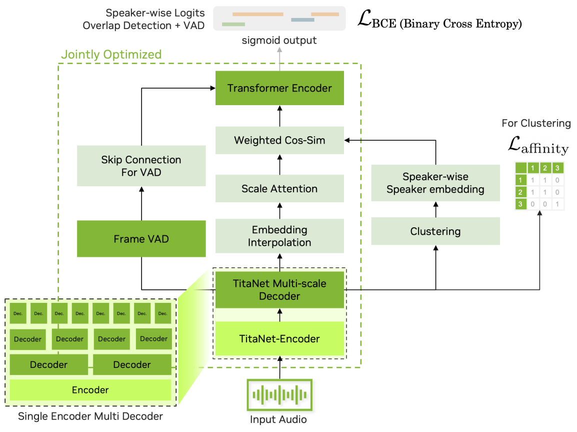

Figure 6: *Data-flow of speaker diarization system.*

the input to the VAD model first undergoes processing by the front-end system, as detailed in Section 2.2. For multi-channel VAD inference, we employ the VAD model trained on a single-channel signal. We exclude channels that have relatively low average VAD probabilities, ranging from 0% to 50%. Subsequently, a maximum operation is applied to the VAD probabilities for each frame, generating the final VAD probability values for the multi-channel audio input.

2.2.2. Multi-channel Diarization Module

Our DASR system is based on the speaker diarization system using the multi-scale diarization decoder (MSDD) proposed in [13]. This system employs a multi-scale embedding approach and utilizes TitaNet [14] speaker embedding extractor. Unlike the system in [13] that uses a multi-layer long short-term memory (LSTM) architecture, we employ a four-layer Transformer architecture with a hidden size of 384. This neural model generates logit values indicating speaker existence. Our diarization model is trained on approximately 3,000 hours of simulated audio mixture data. This data is sourced from the same multi-speaker data simulator used in VAD model training, drawing from the VoxCeleb1&2 [9, 10] and LibriSpeech [15] datasets. Additionally, MUSAN [12] noise is also used for adding additive background noise, primarily focusing on music and broadband noise.

During inference, we employ NME-SC [5] for initial clustering. To leverage the multi-channel input, we concatenate the embedding vectors from each channel to create an elongated embedding vector. Specifically, given M clustered channels, the channel displaying the lowest correlation in the speaker embedding vector is excluded, resulting in $(M-1)$ concatenated embedding vectors. The global speaker clustering result is applied to each channel to produce separate channel-wise diarization outcomes. For multi-scale embedding extraction, we use scale lengths of 3 s, 1.5 s, and 0.5 s with a half-overlap, appli-

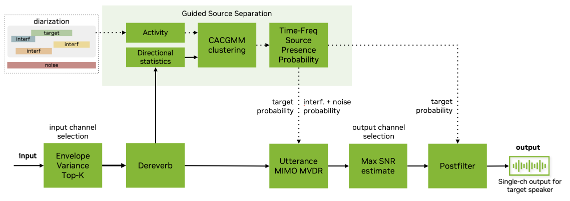

Figure 7: *Multi-channel ASR front-end.*

cable for both clustering and MSDD inference. For clustering, we adjust the multi-scale weight using the following equation for the k-th scale:

$$w(k) = r - \frac{r-1}{K-1}k, \tag{2}$$

where k is an integer scale index, K denotes the number of scales (K=3) and r corresponds to the `r_value`. This `r_value` is parameterized to define the scale weights using a single floating-point number. We optimize the multi-scale weights on the development set by tuning the r value. If $r > 1$, more weight is placed on the longer scale, whereas if $r < 1$ the longer scales are given less weight.

We employ a scale interpolation technique on the smallest scale, achieving a finer speaker-decision resolution of 0.05 seconds. As illustrated in Fig. 4, we select the two segments closest to the center of the interpolated segments and then compute the distance to the center of the neighboring segments. The weights for interpolating the scale to obtain the interpolated embedding vector, $\mathbf{e}$, are determined as follows:

$$d_L = d(t_C, t_L) = |t_C - t_L| \tag{3}$$

$$d_R = d(t_C, t_R) = |t_C - t_R| \tag{4}$$

$$\mathbf{e} = \frac{d_L}{\sqrt{d_L^2 + d_R^2}}\mathbf{e}_L + \frac{d_R}{\sqrt{d_L^2 + d_R^2}}\mathbf{e}_R, \tag{5}$$

where t_L and t_R represent the timestamps of the centers of the two closest segments, while $\mathbf{e}_L$ and $\mathbf{e}_R$ are the two closest speaker embedding vectors on the left and right, respectively. The process of computing interpolated embeddings is executed through a series of matrix-based algebraic multiplications that support batch processing. The MSDD inference window has a local window length T_l of 15 s with a hop length of 3 s. At each inference window, a set of average of speaker embedding vectors is calculated by mixing both the local context (a few tens of seconds, represented by T_l) and global context (a few minutes, represented by T_g). We parameterize the average speaker embedding by using the `global_average_mix_ratio`, α and the `global_average_window_length`, T_g. The average speaker embedding vector of each speaker $\mathbf{E}_S$, is given by:

$$\mathbf{E}_S = \alpha\mathbf{E}_{\text{local},T_l} + (1-\alpha)\mathbf{E}_{\text{global},T_g}, \tag{6}$$

where $\mathbf{E}_{\text{local}}$ and $\mathbf{E}_{\text{global}}$ represent the local and global speaker-profile embedding vectors, respectively. Both vectors have dimension of `(number of scales,embedding dimension,number of speakers)`. The remainder of the training routine is consistent with the system proposed in [13].

2.2.3. *Post-processing of diarization segments*

The final output of the diarization system depicted in 6 is $T \times N_S$ matrix $\mathbf{P}_S$ filled with sigmoid values. Here, N_S represents the maximum number of speakers. In our submitted system, we set N_S=1. The sigmoid values in $\mathbf{P}_S$ are then thresholded using the `sigmoid_threshold`, denoted as τ:

$$P_S[s,i] > \tau, \tag{7}$$

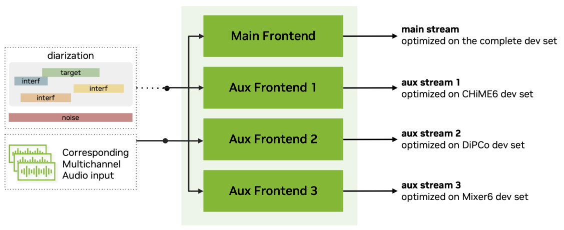
diarization
target
interf
interf
interf
noise
Corresponding Multichannel Audio input
Main Frontend
Aux Frontend 1
Aux Frontend 2
Aux Frontend 3
main stream
optimized on the complete dev set
aux stream 1
optimized on CHiME6 dev set
aux stream 2
optimized on DiPCo dev set
aux stream 3
optimized on Mixer6 dev set

Figure 8: *Multi-stream ASR front-end: multiple audio outputs are generated for each input utterance.*

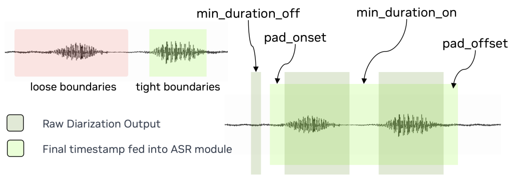

Figure 9: *Diarization post-processing using minimum duration, maximum duration, onset padding and offset padding.*

where s and i are indices for the speaker and the finest-scale segments, respectively. The final diarization segments are derived from the timestamps generated by this thresholding process. Finally, we use majority voting on the channel-specific sigmoid values to generate a single set of diarization segments for each speaker.

As previously demonstrated in [1], the post-processing of diarization segments can significantly influence the performance of both front-end processing and ASR. Thus, we parameterized the post-processing of these diarization segments, as illustrated in Fig. 9. If the segment boundaries are set too tightly, there is a risk of cutting off words; conversely, if they are set too loosely, they may inadvertently include words spoken by the wrong speaker. As depicted, `pad_onset` and `pad_offset` represent the lengths of buffers appended at the start and end of each segment, respectively. Meanwhile, `min_duration_off` and `min_duration_on` are threshold values used for removing short segments and brief silences, respectively.

2.3. Multi-channel ASR front-end

A multi-channel front-end depicted in Fig. 7 based on guided source separation (GSS) is used to extract the target speech signal for a single utterance, as in the baseline system [16]. The multi-channel front-end consists of envelope variance-based channel selection [17, 1], MIMO dereverberation [2] and mask-based MIMO MVDR beamforming [18] for extraction of the target speech signal. Target mask is estimated using GSS with an additional context to prevent permutation [19]. The reference output channel is automatically selected based on maximization of the estimated SNR [19] and the target mask with a minimum gain threshold is used to mask the output reference channel. The front-end is implemented in [3] and optimized to reduce computation time to reduce the time required for parameter optimization of the complete system as described in Section 3.

The main processed audio stream is generated using parameter optimized on the complete CHiME-7 development set and used for Systems 2 and 3 in Tables 1 and 4. In the multi-stream configuration depicted in Fig. 8, we generate three auxiliary processed audio streams using parameters optimized on each of the three subsets in the development set. In this configuration, each utterance is processed using all four front-end setups in parallel and the optimal processed audio stream for the current utterance is selected using an ASR-based confidence

Table 1: *System types and descriptions for main track.*

System	Description
System-1	MC Diarization (cascaded optimization) + Multi-stream Front-End Ensemble + ASR + LM
System-2	MC Diarization + Front-End + ASR + LM (end-to-end optimization)
System-3	MC Diarization (cascaded optimization) + Front-End + ASR-LM parameter optimization

measure. For our System-1 in Table 1, we ensemble the four processed streams with the first preference to main processed audio stream. An auxiliary stream is used if the corresponding ASR segment confidence is greater than the confidence of the main stream by a threshold, which is tuned to achieve the best performance on the development set. The ASR confidence is calculated using the method proposed in [20] with exponentially normalized Tsallis entropy with a temperature of 0.25 and mean aggregation over output tokens.

2.4. Automatic speech recognition

Single-channel audio generated using multi-channel front-end (described in Section 2.3) is transcribed using a 0.6B parameter Conformer-based transducer model [21]. The model was initialized using a publicly available NeMo checkpoint [22]. It was then fine-tuned on the CHiME-7 train set (which includes the CHiME-6 and Mixer6 training subsets) after processing the data through the multi-channel ASR front-end, utilizing ground-truth diarization. This fine-tuning phase lasted for 35,000 updates with a batch size set to 128.

Note that we use the ASR model trained only on CHiME-7 train set for System-A in Table 2 and all three systems in Table 4. Additionally, we also included dev subset data to fine-tune the model used for submission for System-B and System-C in Table 2. All systems use an external language model as described in Section 2.5.

2.5. N-gram language model

The performance of our end-to-end automatic speech recognition model is improved using beam-search algorithm with a language model (LM). We apply a word-piece level N-gram language model with byte-pair-encoding (BPE) tokens using SentencePiece [23, 24] and KenLM [25, 26] toolkits from transcription of CHiME-7 train and dev sets. Token sets of our ASR and LM model were matched. To combine several N-gram models with equal weights we used OpenGrm library [27, 28]. MAES decoding [29] was used for transducer. As expected, the end-to-end model with beam-search decoder and language model performs better than the pure one.

2.6. Text Normalization

We implement basic text normalization rules that correct consistent errors or undesired tokens found in the transcript. We remove all double spaces, unknown symbol `\u2047` from ASR model and replace `aw` with `oh`. These string mappings are validated using the development set. In addition, we employ CHiME-7 scoring for text normalization to produce the final transcription output intended for submission.

3. Hyper-parameter Optimization

For hyper-parameter optimization, we utilize the *Optuna* framework [30]. This framework facilitates the optimization of parameters for black-box systems based on a target metric that assesses system performance. We employ the default optimization algorithm, the tree-structured Parzen estimator [31], aiming to optimize the macro DA-WER on the development set. As illustrated in Fig. 10, our DASR pipeline is treated as a black box,

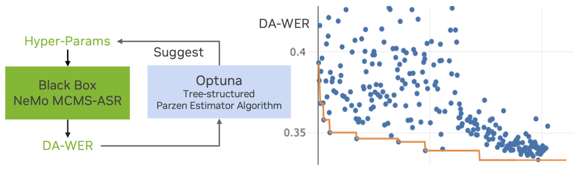

Figure 10: *Optuna optimization loop and scatter plot of trials and DA-WER.*

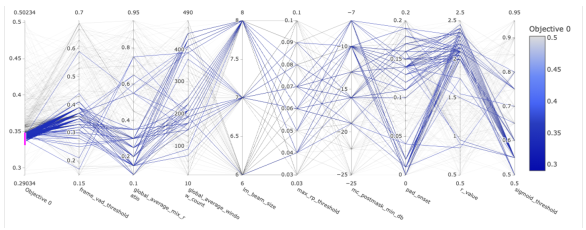

Figure 11: *Parallel Coordinate plot of the major hyper-parameters. The systems showed DA-WER in the range of [33.2, 34.5]% are highlighted.*

given that the optimization does not leverage any prior understanding of the mechanism behind DA-WER computation. We employ two distinct strategies for system optimization: (1) *cascaded* and (2) *end-to-end* optimization. All optimization tasks run using NVIDIA V100 GPUs, and we run five instances of inference sessions on an 8-GPU machine. The entire process takes roughly 9 hours to produce transcriptions for the complete CHiME-7 dev set. Overall, we conduct approximately 2000 trials to identify the best-performing configurations.

3.1. Cascaded Optimization

The cascaded optimization comprises two stages. In the first stage, we optimize the diarization front-end, VAD, diarization module, multi-channel ASR front-end, and the ASR module with greedy decoding. In the second stage, we use the front-end outputs as input to an advanced ASR module equipped with a language model and beam-search decoding. At this stage, we focus on optimizing LM scoring and decoding strategies.

3.2. End-to-End optimization

The end-to-end optimization tunes the hyper-parameters of all modules simultaneously, encompassing LM rescoring and beam search decoding for ASR. To reduce the search space, we fix certain less important hyper-parameters to the best values identified in the previous cascaded optimization phase and optimize those with greater importance and more varied values across the top-performing trials.

3.3. Parameter importance

The hyper-parameters listed below are the top seven that exhibit the most significant importance during the optimization process, sorted in order of their significance:

- `lm_beam_size`: beam search width for LM application
- `r_value`: multi-scale weight scaler r in Eq. (2)
- `global_average_mix_ratio`: α in Eq. (6)
- `global_average_window_length`: T_g in Eq. (6)
- `mc_postmask_min_db`: maximum level of post-mask normalization for multi-channel front end
- `pad_onset`: pad-onset for diarization output segments
- `sigmoid_threshold`: sigmoid threshold τ in Eq. (7)

Additionally, Fig. 11 displays a parallel coordinate plot that highlights the relationships among the major hyper-parameters. It is essential to recognize that the concentration of parameter

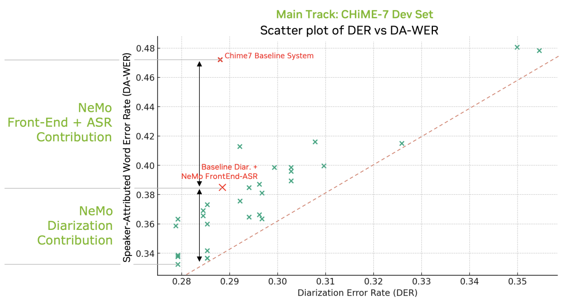

Figure 12: *Scatter plots of DER vs DA-WER*

Table 2: *Sub-track dev set in terms of DA-WER (%).*

	CHiME-6	DiPCo	Mixer6	Macro
System-A	22.8	27.6	12.8	21.0
System-B	16.9	18.7	7.9	14.4
System-C	16.7	18.8	7.9	14.5

Table 3: *Sub-track eval set in terms of DA-WER (%).*

	CHiME-6	DiPCo	Mixer6	Macro
System-A	25.3	25.0	15.8	22.1
System-B	25.6	22.9	15.9	21.4
System-C	25.7	22.4	15.2	21.1

values doesn't directly correlate with parameter importance.

For the sub-track, since the oracle diarization segments are provided, we focus on optimizing the ASR (Section 2.4) and LM (Section 2.5) parameters using Optuna. All three systems for the sub-task are optimized on audio signals produced using the default front-end parameters of the CHiME-7 baseline system. The results of our three systems for the sub-track are presented in Table 2. Note that the WER of System-B and System-C are lower than System-A because their ASR model has been trained on the combined CHiME-7 train and dev subsets. The hyperparameters used for ASR models in the sub-task (System-A/B/C) and main task (System-1/2/3) are separately tuned for the two tasks, and do not share the same sets of parameters.

Table 4 details the systems used in the main track with our diarization system. For System-1 and System-3, we optimize the diarization module, front-end, and ASR simultaneously. Subsequent to this, we select a specific set of parameters for both the diarization and the front-end. For System-3, an ASR model undergoes separate optimization to achieve the lowest DA-WER. For System-2, all modules involved in the DASR task are optimized concurrently through an exhaustive hyperparameter search. It is important to note that the diarization error rate (DER) values in Table 4 are not necessarily optimal. The systems are primarily tuned to minimize the macro-averaged DA-WER, not the DER.

4. Experimental Results

4.1. Evaluation Results

Table 2 and Table 3 display the sub-track results, which are based on the oracle diarization segments. Our best-performing systems exhibit 14.46% on the dev set and 21.1% on the eval set. Note that for the dev set result in the sub-track, we rearrange the splits and include the dev set split in the training set for fine-tuning the ASR model. The tendency observed in the dev set persists in the eval set, where dev-set optimized systems (System-B/C) demonstrate improvement over System-A.

Table 4 and Table 5 display the main-track results, which are based on the oracle diarization segments. The best-

Table 4: *Main track dev set in DER, JER and DA-WER(%).*

		CHiME-6	DiPCo	Mixer6	Macro
System-1	DER	38.6	29.5	15.7	27.9
	JER	39.9	32.6	19.5	30.7
	WER	41.7	40.0	18.0	33.2
System-2	DER	41.6	28.8	17.9	29.5
	JER	43.1	31.7	21.9	32.2
	WER	42.9	39.1	18.0	33.4
System-3	DER	38.6	29.5	15.7	27.9
	JER	39.9	32.6	19.5	30.7
	WER	42.3	40.8	18.1	33.7

* System-3 uses the same diarization system as System-1.

Table 5: *Main track eval set in DER, JER and DA-WER(%).*

		CHiME-6	DiPCo	Mixer6	Macro
System-1	DER	56.1	28.7	18.0	34.3
	JER	56.7	35.7	17.8	36.7
	WER	53.1	34.5	28.0	38.6
System-2	DER	56.0	30.1	20.2	35.4
	JER	55.7	35.6	17.1	36.1
	WER	52.8	36.6	25.9	38.4
System-3	DER	53.5	26.8	15.5	31.9
	JER	57.3	35.4	22.1	38.3
	WER	54.1	35.5	30.3	40.0

performing systems from our submitted systems score 33.2% on the dev set and 38.4% on the eval set. Note that our DER scores are relatively lower than those of other teams or the baseline system because our systems are optimized solely based on the DA-WER value; the DER value is not a factor in selecting the best-performing system. Among our submitted systems, the system with end-to-end hyper-parameter optimization exhibits the lowest DA-WER.

4.2. Ablation study with baseline system

As mentioned in the CHiME-7 challenge documentation [16], the best-performing DA-WER of the baseline system registers at 47.2%, and the DER at 28.8%, on the CHiME-7 *Dev* set. We substitute the diarization of our system with that of the baseline diarization system and assess its performance. We refer to this system as `Baseline Diar.+NeMo FrontEnd-ASR`. By evaluating the performance of this system, we discern the contributions of diarization and the other components of the system, such as the Front-end, ASR, LM, and so on, as depicted in Fig. 12. In Fig. 12, we present a scatter plot depicting the relationship between WER and DER for all the systems we test during the development process. By substituting the baseline diarization with our diarization system, we enhance the DA-WER from 38.4 to 33.22. It is noteworthy that the DER of our best-performing system for *Dev* stands at 27.86%, which is close to the 28.8% of the baseline diarization. However, by utilizing our diarization system, we achieve a significant absolute improvement in DA-WER by 5.2%. We surmise that a contributing factor to this improvement is that the baseline diarization system exhibits a considerably higher confusion error rate compared to our system's diarization. This discrepancy can potentially lead to doubling the WER count, as a word counted as a deletion in one speaker's transcription might be counted as an addition in another speaker's transcription.

4.3. Discussions

The challenge of domain diversity underscores the difficulty of identifying a single optimal configuration due to the unique acoustic and lexical characteristics inherent in each domain. This challenge is further exacerbated by the scarcity of domain-specific training data as our submitted systems are not directly trained on the CHiME-6 development set. When considering the multi-channel front-end, it is essential that processing be seamlessly integrated with diarization. This conclusion is derived from the result we get from end-to-end hyper-parameter optimization. In the realm of ASR techniques, confidence-based ASR ensembling has shown significant promise, although it is not included in the best-performing system. Lastly, pure hyper-parameter optimization alone resulted in over a 16% relative DA-WER reduction. It is crucial to note that a low DER does not necessarily guarantee a low DA-WER. Conversely, a high DER will never result in a low DA-WER. Moreover, even when DER values are similar, the resulting DA-WER can differ significantly. This variation is evident when comparing false alarms and confusion errors between the baseline diarization and the diarization system of our submitted systems.

5. Conclusions

In this paper, we detailed the DASR system of NeMo team for CHiME-7 challange. Our approach enhances the multi-channel speaker diarization and ASR system with dereverberation and channel clustering, allowing for effective handling of the multi-channel input signal. We then employ late-fusion techniques tailored for multi-channel diarization. Furthermore, to boost the performance of the GSS module, we incorporate MIMO dereverberation and MVDR beamforming techniques. Importantly, we optimize all non-differentiable parameters based on DA-WER, ensuring end-to-end inclusion of all modules. All models and codebases discussed in this paper will be made publicly available. Future work can encompass various avenues for system enhancement. First, we aim to explore improved methods to adapt the diarization model, specifically MSDD-based models, to domain-specific data. To counteract overfitting on the limited development dataset, we consider employing a mix of regularization techniques, varied training data, and tailored learning rate controls. Second, we intend to further fine-tune the ASR front-end system to cater not only to spatial cues but also to cues related to speaker identity. We observe that there is potential for refinement in spatial separation, as it occasionally includes speech from the incorrect speaker. Lastly, implementing large language models (LLMs) on top of the decoded ASR output may lead to further reductions in DA-WER by rectifying lexically straightforward errors.

6. References

[1] I. Medennikov *et al.*, "The STC system for the CHiME-6 challenge," in *Proc. CHiME Workshop*, 2020.

[2] T. Yoshioka and T. Nakatani, "Generalization of multi-channel linear prediction methods for blind MIMO impulse response shortening," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 20, no. 10, pp. 2707–2720, 2012.

[3] O. Kuchaiev *et al.*, "NeMo: a toolkit for building AI applications using neural modules," in *Proc. Systems for ML Worshop, NeurIPS*, 2019.

[4] A. J. Muñoz-Montoro, P. Vera-Candeas, and M. G. Christensen, "A coherence-based clustering method for multichannel speech enhancement in wireless acoustic sensor networks," in *Proc. EUSIPCO*, 2021, pp. 1130–1134.

[5] T. J. Park *et al.*, "Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap," *IEEE Signal Processing Letters*, vol. 27, pp. 381–385, 2020.

[6] NVIDIA, "Speech to frame label," https://github.com/NVIDIA/NeMo/blob/main/examples/asr/speech_classification/speech_to_frame_label.py, [Online; accessed July 17, 2023].

[7] S. Watanabe *et al.*, "CHiME-6 Challenge: Tackling Multispeaker Speech Recognition for Unsegmented Recordings," in *Proc. CHiME-6 Workshop*, 2020, pp. 1–7.

[8] NVIDIA, "Speech data simulator," https://github.com/NVIDIA/NeMo/tree/main/tools/speech_data_simulator, [Online; accessed July 17, 2023].

[9] A. Nagrani *et al.*, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[10] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[11] D. S. Park *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[12] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[13] T. J. Park *et al.*, "Multi-scale speaker diarization with dynamic scale weighting," *arXiv preprint arXiv:2203.15974*, 2022.

[14] N. R. Koluguri, T. Park, and B. Ginsburg, "TitaNet: Neural model for speaker representation with 1d depth-wise separable convolutions and global context," in *Proc. ICASSP*, 2022, pp. 8102–8106.

[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[16] S. Cornell *et al.*, "The CHiME-7 DASR Challenge: Distant Meeting Transcription with Multiple Devices in Diverse Scenarios," in *Proc. CHiME Workshop*, 2023.

[17] M. Wolf and C. Nadeu, "Channel selection measures for multimicrophone speech recognition," *Speech Comm.*, vol. 57, pp. 170–180, 2014.

[18] M. Souden, J. Benesty, and S. Affes, "On optimal frequency-domain multichannel linear filtering for noise reduction," *IEEE Trans. on Audio, Speech, and Lang. Process.*, vol. 18, no. 2, pp. 260–276, 2009.

[19] C. Boeddeker *et al.*, "Front-end processing for the CHiME-5 dinner party scenario," in *Proc. CHiME-5 Workshop*, 2018.

[20] A. Laptev and B. Ginsburg, "Fast entropy-based methods of word-level confidence estimation for end-to-end automatic speech recognition," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2023, pp. 152–159.

[21] A. Gulati *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020.

[22] NVIDIA, "Conformer Transducer XL," https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_transducer_xlarge, [Online; accessed July 17, 2023].

[23] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. EMNLP: System Demonstrations*, Nov. 2018, pp. 66–71.

[24] Google, "SentencePiece," https://github.com/google/sentencepiece, [Online; accessed July 17, 2023].

[25] K. Heafield, "KenLM: Faster and smaller language model queries," in *Proc. Workshop on Statistical Machine Translation*, Jul. 2011, pp. 187–197.

[26] ——, "kenlm," https://github.com/kpu/kenlm, [Online; accessed July 17, 2023].

[27] B. Roark *et al.*, "The OpenGrm open-source finite-state grammar software libraries," in *Proc. ACL 2012 System Demonstrations*, Jul. 2012, pp. 61–66.

[28] ——, "OpenGrm NGram Library," https://www.opengrm.org/twiki/bin/view/GRM/NGramLibrary, [Online; accessed July 17, 2023].

[29] J. Kim, Y. Lee, and E. Kim, "Accelerating RNN transducer inference via adaptive expansion search," *IEEE Signal Process. Letters*, vol. 27, pp. 2019–2023, 2020.

[30] T. Akiba *et al.*, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2019.

[31] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems," in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 533–541.