INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Improving End-to-End Neural Diarization Using Conversational Summary Representations

Samuel J. Broughton, Lahiru Samarakoon

Fano Labs, Hong Kong SAR, China
{samuel.broughton,lahiru}@fano.ai

Abstract

Speaker diarization is a task concerned with partitioning an audio recording by speaker identity. End-to-end neural diarization with encoder-decoder based attractor calculation (EEND-EDA) aims to solve this problem by directly outputting diarization results for a flexible number of speakers. Currently, the EDA module responsible for generating speaker-wise attractors is conditioned on zero vectors providing no relevant information to the network. In this work, we extend EEND-EDA by replacing the input zero vectors to the decoder with learned conversational summary representations. The updated EDA module sequentially generates speaker-wise attractors based on utterance-level information. We propose three methods to initialize the summary vector and conduct an investigation into varying input recording lengths. On a range of publicly available test sets, our model achieves an absolute DER performance improvement of 1.90 % when compared to the baseline.

Index Terms: end-to-end neural diarization (EEND), EEND-EDA, conversational summary vector

1. Introduction

Speaker Diarization is a task concerned with determining the number of speakers and their respective speech activities for a given input audio signal, often referred to as the problem of *"who spoke when?"* [1]. A diarization system is considered to be robust if it can perform well when dealing with overlapping speech segments, long form audio and an arbitrary number of speakers across a range of acoustic domains [2, 3].

Traditional methods employed to achieve diarization have involved clustering-based approaches within a modular pipeline structure [4, 5]. Given an input audio signal, speaker active frames are detected with a voice activity detection module (VAD). Speaker embedding vectors are extracted from speaker active frames and clustered so that segments belonging to the same speaker can be labelled. The main disadvantage with clustering-based approaches is that they cannot handle overlapping speech, an inherent characteristic of real world conversational data [2, 6]. Some methods have been designed to address this problem [7, 8], however, this substantially increases the complexity of the solution and inter-module dependencies.

End-to-end neural-based diarization systems directly solve the overlapping speech issue, simplify the overall design of the diarization pipeline and generally perform better than traditional approaches. End-to-end neural diarization (EEND) directly outputs diarization results by treating the task as a multi-label classification problem [9]. EEND-based models require permutation invariant training (PIT), as they currently predict diarization results without taking into consideration speaker order [10]. An encoder-decoder attractor calculation module (EDA) can replace the classification head in EEND to flexibly determine the number of speakers in a given utterance [11, 12]. The EDA module is responsible for creating speaker-wise attractor representations for diarization results calculation and speaker existence prediction.

However, the LSTM-based EDA module has limitations. LSTM networks are prone to vanishing gradients when handling long sequences such as recordings used for training in diarization. This means for recordings containing a higher number of active speakers, the EDA module will struggle to produce well-separated attractor representations. To tackle this problem, the original authors shuffled input embeddings to the LSTM encoder [12], and studied the use of both global and local attractor calculation [13]. One key issue is that the EDA LSTM decoder is expected to sequentially generate speaker-wise attractor representations from only the last hidden and cell states of the LSTM encoder. The input zero vector to the decoder contains no relevant information to the network and is common to all recordings processed by the model. By estimating a more meaningful representation the model could perform better for recordings containing a higher number of active speakers.

This work presents a feature-based approach to enhance attractor representations in the EDA module of EEND-EDA for better diarization performance in recordings containing a higher number of active speakers. Inspired by the special classification token in BERT [14], we introduce a summary vector to the first frame of the subsampled acoustic input feature sequence to learn a conversational summary representation. This learned summary replaces the input zero vector to the EDA module previously used, giving the LSTM decoder additional utterance-level information. Incorporating conversational summary representations to the EDA module achieves a lower diarization error rate (DER) when compared to the baseline, particularly for recordings with a higher number of active speakers.

In Section 2 we revisit EEND-EDA, Section 3 outlines the usage of the summary vector, Section 4 explains the experimental conditions, Section 5 discusses the results, Section 6 concludes the paper.

2. EEND-EDA

EEND is the current state-of-the-art end-to-end diarization architecture used by the community [9, 3, 15, 16, 17]. An EDA module was introduced to allow EEND to handle an arbitrary number of speakers [11, 12, 18, 19, 20].

The main training objective of EEND-based diarization is to learn a mapping function that outputs the likelihood of speech activities for multiple speakers given an input sequence $X \in \mathbb{R}^{D \times T}$. Here, X is a log-scaled Mel-filterbank acoustic input feature sequence with dimension D and length T. Out-

10.21437/Interspeech.2023-2401

put embedding $E \in \mathbb{R}^{D \times T}$ is produced by passing X as input to a Transformer-based encoder [21, 22]. Time shuffled embedding E is input to an LSTM-based EDA module to predict speaker-wise attractors $A \in \mathbb{R}^{D \times (S+1)}$, where S is the number of speakers.

Diarization results are calculated by a sigmoid function operating on the element-wise multiplication of A and E. This outputs posterior probabilities $(\mathbf{p}_t)_{t=1}^{T}$ for the speech activities of multiple speakers across all frames. Where $\mathbf{p}_t := [p_{t,1}, ..., p_{t,S}] \in (0,1)^S$ are the posterior probabilities for S speakers at frame t. These posteriors are without conditions to decide the order of the speakers. During inference, the scalar value of $p_{t,s}$ is used with a threshold value of 0.5 to determine whether speaker s should be labelled as active or not at frame t.

Diarization Loss. A permutation free objective is used to optimize the model by calculating the loss for all possible speaker assignments between diarization predictions and groundtruth labels $(\mathbf{y}_t)_{t=1}^{T}$ [10]. Where $\mathbf{y}_t := [y_{t,1}, ..., y_{t,S}] \in \{0,1\}^S$ are the groundtruth labels for all speakers at frame t. Here, $y_{s,t} = 1$ and $y_{s,t} = 0$ denotes whether speaker s is active or not at t. The minimum loss is then taken for backpropagation and can be defined as:

$$\mathcal{L}_{diar} = \frac{1}{TS} \min_{i \in \text{perm}(1,...,S)} \sum_{t=1}^{T} H(\mathbf{p}_t^i, \mathbf{y}_t), \tag{1}$$

where $\text{perm}(1, ..., S)$ and $\mathbf{p}_t^i$ resemble the set of permutations for all speakers and the permuted posterior labels at frame t, respectively. Here, $H(\cdot, \cdot)$ is the binary cross entropy.

Attractor Existence Loss. The output hidden and cell states of the EDA LSTM encoder h^{enc} are used to initialize the hidden and cell states of the EDA LSTM decoder h^{dec}. Speaker-wise attractors are generated from input zero vectors:

$$\mathbf{h}_s^{\text{dec}}, \mathbf{c}_s^{\text{dec}} = h^{\text{dec}}(\mathbf{0}, \mathbf{h}_{s-1}^{\text{dec}}, \mathbf{c}_s^{\text{dec}}) \qquad (s = 1, ..., S), \tag{2}$$

where hidden state $\mathbf{h}_s^{\text{dec}}$ corresponds to speaker s's attractor. Attractor existence posterior probabilities are calculated by inputting A to a fully connected layer followed by sigmoid activation. Attractor existence loss $\mathcal{L}_{exist}$ is calculated using equation (19) in [12].

The **full training objective** of EEND-EDA can be defined as:

$$\mathcal{L} = \mathcal{L}_{diar} + \alpha \mathcal{L}_{exist}, \tag{3}$$

where α is a hyperparameter.

3. Summary Vector for EEND-EDA

This Section explains the feature-based enhancement to the EDA module of EEND-EDA [11], by learning a conversational summary representation. Figure 1 presents the proposed network architecture.

3.1. Summary vector estimation

The original encoder of EEND-EDA takes an input sequence X and outputs embedding E. Inspired by how the special `[CLS]` token is used in BERT [14], we modify the encoder to additionally estimate a summary representation $\hat{\mathbf{u}} \in \mathbb{R}^D$ of the original input sequence:

$$\hat{\mathbf{u}}, E = \text{enc}(X). \tag{4}$$

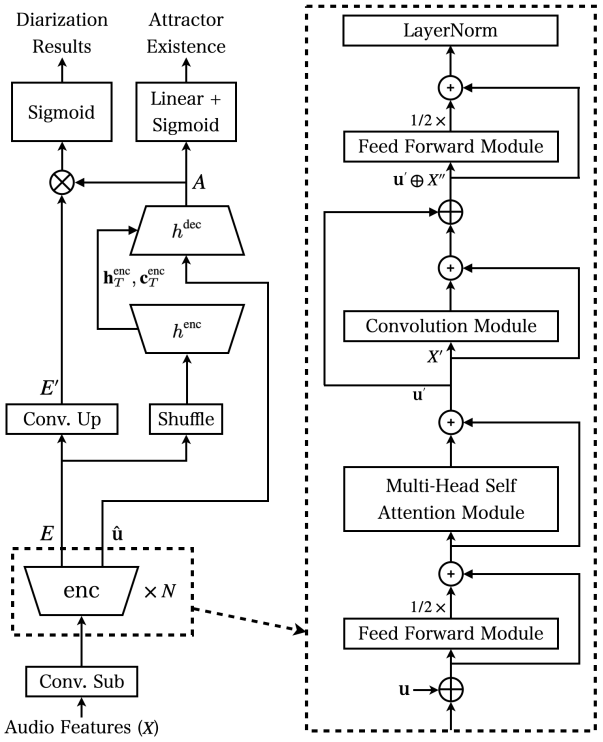

Figure 1: *Proposed network architecture for EEND-EDA with special summary vector representation. The network is similar to EEND-EDA with convolutional subsampling and upsampling [20]. The dashed box on the right shows the modification to Conformer encoder when used with a special summary vector representation.*

E is input to the rest of EEND-EDA as usual [20]. We replace the original input zero vectors to h^{dec} in (2), such that each estimated speaker-wise attractor is conditioned with $\hat{\mathbf{u}}$:

$$\mathbf{h}_s^{\text{dec}}, \mathbf{c}_s^{\text{dec}} = h^{\text{dec}}(\hat{\mathbf{u}}, \mathbf{h}_{s-1}^{\text{dec}}, \mathbf{c}_s^{\text{dec}}) \qquad (s = 1, ..., S). \tag{5}$$

At each iteration the LSTM decoder now sees a conversational summary representation of the input audio sequence. h^{enc} remains unchanged from the original design, as seen in equation (16) of [12].

3.2. Summary vector formulation

We outline three different methods to initialize conversational summary representation $\mathbf{u}$ of the encoder:

- Average pooling is used to summarize the mean presence of features in X.
- Max pooling is used to summarize the most activated presence of features in X.
- A randomly initialized parameter of size D is added to the encoder network so that it's gradients can be updated by the training objective.

3.3. Conformer encoder modification

Summary vector $\mathbf{u}$ is concatenated with input feature sequence X for input to modules of the Conformer encoder [22], such that the first frame of every sequence corresponds to a special summary representation.

Table 1: *Datasets used at various experimental stages.*

Stage	Dataset	#Speakers	#Mixtures
Pre-training 1	LibriSpeech (β=2)	2	100,000
Pre-training 2	LibriSpeech (β=2, 2, 5, 9)	1 - 4	400,000
Fine-tuning	VoxConverse Dev.	1 - 20	216
	DIHARD III Dev. (Train)	1 - 10	162
	DIHARD III Dev. (Val.)	1 - 10	41
	MagicData-RAMC Train	2	289
	AMI Mix	3 - 5	136
	AMI SDM1	3 - 5	135
Testing	VoxConverse Evaluation Set	1 - 21	259
	DIHARD III Evaluation Set	1 - 9	232
	MagicData-RAMC Test	2	43
	AMI Mix	3, 4	16
	AMI SDM1	3, 4	16

The dashed box on the right of Figure 1 shows our modification to the Conformer encoder model architecture when using an input with a special summary representation. The structure of the encoder block is the same, using two feed-forward modules (FFN) in the style of Macaron-Net [23], a Multi-Head Self Attention Module (MHSA), Convolution Module (Conv) and layer normalization (LayerNorm). However, the hidden summary vector skips the Convolution module to ensure it maintains a global representation. The hidden feature sequence passes through the Convolution module with a residual connection. The hidden summary vector is concatenated back with the hidden feature sequence before input to the second feed-forward module.

Mathematically, the outputs $\hat{\mathbf{u}}, E$ of a single modified Conformer block with summary vector $\mathbf{u}$ and input X is:

$$\begin{aligned} \tilde{\mathbf{u}}, \tilde{X} &= (\mathbf{u} \oplus X) + \frac{1}{2}\text{FFN}(\mathbf{u} \oplus X) \\ \mathbf{u}', X' &= (\tilde{\mathbf{u}} \oplus \tilde{X}) + \text{MHSA}(\tilde{\mathbf{u}} \oplus \tilde{X}) \\ X'' &= X' + \text{Conv}(X') \\ \hat{\mathbf{u}}, E &= \text{LayerNorm}((\mathbf{u}' \oplus X'') + \frac{1}{2}\text{FFN}(\mathbf{u}' \oplus X'')), \end{aligned} \tag{6}$$

where the $\oplus$ notation denotes the concatenation of tensors.

4. Experiments

4.1. Data

Table 1 presents the datasets used across all experiments for each stage of training and evaluation. LibriSpeech [24][1] recordings are used for both pre-training stages and are segmented using WebRTC VAD[2]. The 960 hours LibriSpeech training set is used to simulate mixtures for 1, 2, 3 and 4 speakers by the protocol defined in [11][3]. Where β is used to control the silence duration and overlap ratio.

A number of public datasets covering a wide range of acoustic environments were used for fine-tuning. This included the VoxConverse [2][4] and DIHARD III [1, 25] Development datasets, MagicData-RAMC training set [26] and the AMI Mix and SDM1 training sets [27].

VoxConverse is a real world conversational dataset extracted from YouTube videos. Both the development set (20.3 hours) and evaluation set (43.5 hours) contains a significant proportion of overlapping speech. The DIHARD III dataset is a collection of various challenging datasets from 11 domains with different recording equipment and environments. The DIHARD III Development set, consisting of 203 mixtures, was split into training and validation sets by a ratio of 80%:20% per domain. Magic-RAMC is a high-quality Mandarin conversational speech dataset recorded on mobile phones by native speakers. The AMI Meeting Corpus contains 100 hours of recordings for close-talking (Mix) and far-field (SDM1) microphones in three rooms with varying acoustic properties in English with mostly non-native speakers.

The VoxConverse and DIHARD III Evaluation sets, Magic-RAMC test set and both AMI Mix and SDM1 test sets are used for evaluation.

4.2. Experimental setup

The baseline model is EEND-EDA with no use of summary vector representations. For all models, the architecture of the encoder consisted of four stacked Conformer blocks ($N = 4$) each with four attention heads. The Conformer encoder made use of Macaron style [23] feed-forward layers with 1024 hidden units and outputted 256-dimensional frame-wise embeddings. No positional embeddings were used. Inputs to the model were 23-dimensional log Mel-filterbanks with a frame length of 25 ms and a frame shift of 10 ms. Input features passed through a 10-fold convolutional subsampling module (Conv. Sub) consisting of two convolutional layers with kernel sizes $\{3, 5\}$ and stride $\{2, 5\}$. The convolutional upsampling module (Conv. Up) used two transposed convolutional layers with batch normalisation and ReLU activation. Here, kernel sizes $\{3, 5\}$, strides $\{2, 5\}$ and output padding $\{1, 0\}$ are used. We use the additive margin penalty on diarization results only and set m to 0.35 [20].

Each model was pre-trained for two stages on the LibriSpeech simulated mixtures. In the first stage of pre-training each model was trained on mixtures containing only 2 speakers for 100 epochs. In the second stage each model was then trained for 25 epochs on a concatenation of mixtures containing 1 to 4 speakers. Models were adapted for a total of 500 epochs during the fine-tuning stage.

The Adam optimizer [28] and Noam scheduler [21] with 100,000 warm-up steps was used during pre-training on simulated mixtures. At fine-tuning the Adam optimizer was used with a fix learning rate of 1×10^{-5}. Due to the time complexity of PIT, the model was trained to output the four most dominant speakers. Attractor existence loss $\mathcal{L}_{exist}$ was used to update only the EDA module by cutting the computational graph of all inputs to the EDA LSTM encoder to disable back propagation to the preceding layers.

For all stages of training a batch size of 64 was used. Unless specified, input recording lengths were set to 5000. Chunk shuffling was also used at fine-tuning [19]. All models were trained on one GeForce RTX 3090 GPU for recording lengths of 5000. This included $\sim$ 94 hours for pre-training and $\sim$ 22 hours for fine-tuning. Up to four GPUs were used when increasing recording lengths to 20000 frames. Each model used 8.1M parameters. The evaluation metric is the DER with no collar tolerance. DER is composed of missed speech, false alarm and speaker error metrics [4].

[1] https://www.openslr.org/12/

[2] https://www.github.com/staplesinLA/denoising_DIHARD18/

[3] https://github.com/hitachi-speech/EEND/

[4] https://www.robots.ox.ac.uk/~vgg/data/voxconverse/

Table 2: *DER (%) results of different methods for initialising summary vector across all test datasets combined.*

Model	NS2	NS3	NS4	NS2 to NS4	NS2 to NS9
Baseline	12.01	20.27	28.50	17.89	22.40
SR-AvgPool	**11.99**	19.96	28.62	17.89	22.26
SR-MaxPool	12.74	20.24	26.52	17.71	21.38
SR-Learned	12.46	**19.65**	**24.49**	**16.86**	**20.50**

Table 3: *DER (%) results for each test set.*

Dataset	Model	NS2	NS3	NS4	NS2 to NS4
DIHARD III	Baseline	10.32	23.62	34.29	**14.09**
	SR-Learned	10.90	25.64	34.95	14.82
VoxConverse	Baseline	9.23	19.60	26.15	16.59
	SR-Learned	8.66	14.59	15.34	**11.94**
MagicData-RAMC	Baseline	14.37	-	-	**14.37**
	SR-Learned	14.94	-	-	14.94
AMI Mix	Baseline	-	15.81	21.65	21.01
	SR-Learned	-	16.36	19.06	**18.76**
AMI SDM1	Baseline	-	17.85	34.53	32.72
	SR-Learned	-	17.65	30.12	**28.76**

5. Results

5.1. Results on summary vector initialisation

The first experiment measured the DER performance of the baseline to three models that made use of conversational summary representations. This compared the different methods described in Section 3.2 to initialize the summary vector.

Table 2 presents the results across all test datasets combined. For recordings containing two to four active speakers (NS2 to NS4), we observe a relative DER improvement of 1.03 % using learned summary representations (SR-Learned) when compared to the baseline. Most gains were made in recordings with four active speakers (NS4). Initialising summary vector by applying average pooling (SR-AvgPool) or max pooling (SR-MaxPool) to the subsampled input sequence shows comparable performance to the baseline. SR-Learned further improves the absolute DER by 1.90 % when compared to the baseline for recordings with two to nine active speakers (NS2 to NS9).

Table 3 shows a breakdown of results for each test dataset. On average, SR-Learned outperforms the baseline on VoxConverse and both AMI test sets, whilst performing comparably with the baseline on other test sets. The most gains can be observed in recordings with four active speakers.

Table 4: *DER (%) results for all test datasets combined when varying input recording lengths during fine-tuning.*

Model	NS2	NS3	NS4	NS2 to NS4	NS2 to NS9
Baseline 50s	**12.01**	20.27	28.50	17.89	22.40
Baseline 100s	12.97	19.39	25.62	17.49	21.33
Baseline 150s	13.03	**16.97**	25.79	17.36	21.22
Baseline 200s	13.11	23.67	25.02	17.77	21.50
SR-Learned 50s	12.46	19.65	24.49	16.86	20.50
SR-Learned 100s	12.35	18.99	**22.24**	16.03	**19.74**
SR-Learned 150s	11.28	19.02	23.76	15.87	20.08
SR-Learned 200s	11.87	18.59	22.38	**15.75**	20.02

Table 5: *Dot-product similarity of output embeddings and attractors, split by ground-truth labels. Where labels 1 - 4 refer to frames where individual speakers are active. Labels 5 and 6 denote overlapping and silent frames, respectively.*

(a) Baseline

Labels \ Attractors	1	2	3	4
1	**5.4**	-6.6	-4.2	-5.8
2	-6.6	**3.2**	-3.7	-5.0
3	1.7	-4.9	**-1.8**	-4.1
4	-5.7	-1.8	-5.0	**1.4**
5	0.9	-0.2	-2.1	-0.3
6	-23.9	-13.1	-12.3	-14.9

(b) SR-Learned

Labels \ Attractors	1	2	3	4
1	**6.7**	-9.2	-6.6	-7.5
2	-9.5	**8.9**	-8.0	-7.4
3	-6.6	-7.4	**5.0**	-7.3
4	-9.9	-10.2	-7.9	**8.3**
5	-1.9	-1.7	-1.4	0.7
6	-18.5	-27.2	-14.5	-10.0

5.2. Results on varying input recording lengths

By increasing the input recording lengths during fine-tuning the model was able to train on sequences containing a higher number of active speakers, subsequently increasing the proportion of speakers the EDA module was exposed to.

Table 4 shows the results for the baseline and SR-Learned models when varying input recording lengths during fine-tuning. Results for recording lengths of 5000 frames (50s) are copied from Table 3. Both models achieve lower a DER for three and four active speakers, whilst the baseline degrades in performance for recordings with two active speakers. The best performing baseline uses recording lengths of 15000 frames (150s). SR-Learned improves this with an absolute DER improvement of 1.49 % for two to four active speakers. SR-Learned yields greater improvements when increasing the length of input recordings, showing an absolute DER improvement of 1.11 % for two to four active speakers when compared to the result for 50s. Improvements for two to nine active speakers could be negligible due to the model only being trained to diarize up to four active speakers.

5.3. Insights on the behavior of EDA

To gain more understanding about the behaviour of EDA, we investigate the dot-product similarity between the output upsampled embeddings and attractors for both the baseline and SR-Learned models. Table 5 presents the results for a four speaker mixture from the AMI Mix evaluation set. The speaker frames, overlapping frames and silent frames were assigned based on the ground-truth labels. Attractors for both models are seen to be well separated from embedding frames related to silence. SR-Learned attractors are shown to be more separated from frames not relating to their represented speaker when compared to the baseline.

6. Conclusion

In this work, we introduced conversational summary representations for end-to-end neural diarization with encoder-decoder based attractor calculation. Of the three methods investigated for initializing the summary vector, it was found that an additional parameter to the encoder network yielded the most gains. Further improvements were observed when increasing input recording lengths. Adopting the learned summary representation, the model demonstrated an absolute DER performance improvement of 1.90 % (for recording lengths of 5000 frames) for two to nine active speakers when compared to the baseline.

7. References

[1] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The third dihard diarization challenge," *arXiv preprint arXiv:2012.01477*, 2020.

[2] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, "Spot the conversation: speaker diarisation in the wild," *arXiv preprint arXiv:2007.01216*, 2020.

[3] K. Kinoshita, M. Delcroix, and N. Tawara, "Advances in integration of end-to-end neural and clustering-based diarization for real conversational speech," *arXiv preprint arXiv:2105.09040*, 2021.

[4] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker diarization: A review of recent research," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 2, pp. 356–370, 2012.

[5] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A review of speaker diarization: Recent advances with deep learning," *Computer Speech & Language*, vol. 72, p. 101317, 2022.

[6] F. Yu, S. Zhang, Y. Fu, L. Xie, S. Zheng, Z. Du, W. Huang, P. Guo, Z. Yan, B. Ma *et al.*, "M2met: The icassp 2022 multi-channel multi-party meeting transcription challenge," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6167–6171.

[7] M. Diez, F. Landini, L. Burget, J. Rohdin, A. Silnova, K. Zmolíková, O. Novotnỳ, K. Veselỳ, O. Glembek, O. Plchot *et al.*, "But system for dihard speech diarization challenge 2018." in *Interspeech*, 2018, pp. 2798–2802.

[8] K. Kinoshita, M. Delcroix, and N. Tawara, "Integrating end-to-end neural and clustering-based diarization: Getting the best of both worlds," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7198–7202.

[9] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 296–303.

[10] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 241–245.

[11] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and K. Nagamatsu, "End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors," *arXiv preprint arXiv:2005.09921*, 2020.

[12] S. Horiguchi, Y. Fujita, S. Watanabe, Y. Xue, and P. García, "Encoder-decoder based attractor calculation for end-to-end neural diarization," *arXiv preprint arXiv:2106.10654*, 2021.

[13] S. Horiguchi, S. Watanabe, P. García, Y. Xue, Y. Takashima, and Y. Kawaguchi, "Towards neural diarization for unlimited numbers of speakers using global and local attractors," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 98–105.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[15] A. Khare, E. Han, Y. Yang, and A. Stolcke, "Asr-aware end-to-end neural diarization," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8092–8096.

[16] Y. Ueda, S. Maiti, S. Watanabe, C. Zhang, M. Yu, S.-X. Zhang, and Y. Xu, "Eend-ss: Joint end-to-end neural speaker diarization and speech separation for flexible number of speakers," *arXiv preprint arXiv:2203.17068*, 2022.

[17] Y. Yu, D. Park, and H. K. Kim, "Auxiliary loss of transformer with residual connection for end-to-end speaker diarization," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8377–8381.

[18] S. Horiguchi, N. Yalta, P. Garcia, Y. Takashima, Y. Xue, D. Raj, Z. Huang, Y. Fujita, S. Watanabe, and S. Khudanpur, "The hitachi-jhu dihard iii system: Competitive end-to-end neural diarization and x-vector clustering systems combined by dover-lap," *arXiv preprint arXiv:2102.01363*, 2021.

[19] T.-Y. Leung and L. Samarakoon, "End-to-end speaker diarization system for the third dihard challenge system description," 2021.

[20] ——, "Robust end-to-end speaker diarization with conformer and additive margin penalty." in *Interspeech*, 2021, pp. 3575–3579.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[23] Y. Lu, Z. Li, D. He, Z. Sun, B. Dong, T. Qin, L. Wang, and T.-Y. Liu, "Understanding and improving transformer from a multi-particle dynamic system point of view," *arXiv preprint arXiv:1906.02762*, 2019.

[24] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[25] N. Ryant, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "Third dihard challenge evaluation plan," *arXiv preprint arXiv:2006.05815*, 2020.

[26] Z. Yang, Y. Chen, L. Luo, R. Yang, L. Ye, G. Cheng, J. Xu, Y. Jin, Q. Zhang, P. Zhang *et al.*, "Open source magicdata-ramc: A rich annotated mandarin conversational (ramc) speech dataset," *arXiv preprint arXiv:2203.16844*, 2022.

[27] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal *et al.*, "The ami meeting corpus: A pre-announcement," in *Machine Learning for Multimodal Interaction: Second International Workshop, MLMI 2005, Edinburgh, UK, July 11-13, 2005, Revised Selected Papers 2*. Springer, 2006, pp. 28–39.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.