ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095372

SUPERVISED HIERARCHICAL CLUSTERING USING GRAPH NEURAL NETWORKS FOR SPEAKER DIARIZATION

Prachi Singh , Amrit Kaul, Sriram Ganapathy

LEAP Lab, Electrical Engineering, Indian Institute of Science,Bangalore.

prachisingh@iisc.ac.in

ABSTRACT

Conventional methods for speaker diarization involve windowing an audio file into short segments to extract speaker embeddings, followed by an unsupervised clustering of the embeddings. This multi-step approach generates speaker assignments for each segment. In this paper, we propose a novel Supervised HierArchical gRaph Clustering algorithm (SHARC) for speaker diarization where we introduce a hierarchical structure using Graph Neural Network (GNN) to perform supervised clustering. The supervision allows the model to update the representations and directly improve the clustering performance, thus enabling a single-step approach for diarization. In the proposed work, the input segment embeddings are treated as nodes of a graph with the edge weights corresponding to the similarity scores between the nodes. We also propose an approach to jointly update the embedding extractor and the GNN model to perform end-to-end speaker diarization (E2E-SHARC). During inference, the hierarchical clustering is performed using node densities and edge existence probabilities to merge the segments until convergence. In the diarization experiments, we illustrate that the proposed E2E-SHARC approach achieves 53% and 44% relative improvements over the baseline systems on benchmark datasets like AMI and Voxconverse, respectively.

***Index Terms*—** Supervised Hierarchical Clustering, Graph Neural Networks, Speaker Diarization.

1. INTRODUCTION

Speaker Diarization (SD) is the task of segmenting an audio file based on speaker identity. The conventional approach for the task of SD involves multiple steps. In the first step, the audio is windowed into short segments (1-2 s) and fed to a speaker embedding extractor (for example, x-vectors) [1, 2]. In a subsequent step, these speaker embeddings are clustered based on similarity scores computed using methods like Probabilitic Linear Discriminant Analysis (PLDA) scoring [3, 4]. The most common clustering approach is the agglomerative hierarchical clustering (AHC) [5], which merges two clusters at each time step based on similarity scores until the required number of clusters/speakers are attained. Other approaches involve spectral clustering (SC) [6], k-means clustering [7] and graph based clustering [8, 9].

Recently, the end-to-end neural diarization [10, 11] approaches involving transformers have proved effective in handling overlaps. However, due to the difficulty in handling more than 4 speakers, pairwise metric learning loss is proposed recently [12]. A graph-based agglomerative clustering called path integral clustering proposed by Zhang et al. [8] is shown to outperform other clustering approaches on CALLHOME and AMI datasets [9]. Similarly, metric learning approaches are introduced in [6, 13] to improve the speaker similarity scores. In a recent work, Singh et al. [9, 14] introduced self-supervised metric learning using clustering output labels as the pseudo labels for model training.

Most of the previous approaches for diarization are trained to improve the similarity scores. However, they still use an unsupervised clustering algorithm to obtain the final labels. We hypothesize that this limits their performance as they are not trained with the clustering objectives. On the other hand, EEND models require a large amount of data and hundreds of hours of training. We propose a simple approach to SD which is not data intensive and can handle large number of speakers (more than 7) while training and evaluation. The approach is called as Supervised HierArchical gRaph Clustering algorithm (SHARC). Our work is inspired by the work on image clustering by Xing et al. [15]. The major contributions are:

1. Introducing supervised hierarchical clustering using Graph Neural Networks (GNN) for diarization.
2. Developing the framework for joint representation learning and clustering using supervision.
3. Achieving state-of-the-art performance on two benchmark datasets.

2. RELATED WORK AND BACKGROUND

GCN: Wang et al. [16] proposed GNN for metric learning. The inputs to the model are x-vectors/d-vectors and the PLDA similarity score. The output of the model is a probability score of whether two nodes are connected or not. The Graph Convolution Network (GCN) [17], the most common variant of GNNs, are used in [18] for semi-supervised training using clustering output as "pseudo-labels".

GraphSAGE: The GCN is inherently transductive and does not generalize to unseen nodes. The GraphSAGE [19], another variant of GNN, is a representation learning technique suitable for dynamic graphs. A function is learnt that generates embeddings by sampling and aggregating features from a node's local neighborhood. The aggregate function outputs a single neighborhood embedding by taking a weighted average of each neighbor's embedding. The update equations are as follows:

$$\begin{aligned} h^l_{N(v)} &= \text{Aggregate}(h^{l-1}_u, \forall u \in N(v)) \\ h^l_v &= \sigma(\boldsymbol{W}^l[h^{l-1}_v; h^l_{N(v)}]) \end{aligned} \quad (1)$$

where, $N(v)$ is the neighborhood of node v, $h^l_{N(v)}$ and h^l_v are the latent representations of $N(v)$ and node v respectively, at layer $l \in \{1, ..., L\}$. The σ is the activation function and $\boldsymbol{W}^l$ is the weight matrix. h^l_v is the hidden representation of node v.

This work was supported by the grants from the British Telecom Research Center.

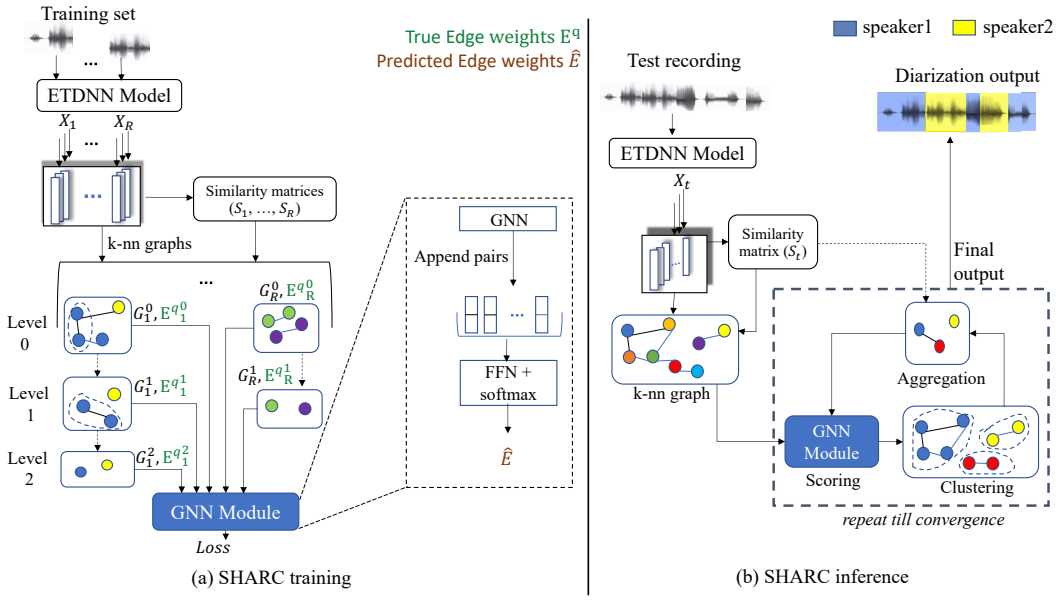

Fig. 1. Block diagram of proposed SHARC method. The ETDNN model and GNN are the extended time delay network model for x-vector extraction and the graph neural network for score prediction. FFN stands for feed forward network. The left side (a) shows the training steps and the right side (b) shows the inference steps.

3. PROPOSED APPROACH

The Supervised HierArchical gRaph Clustering algorithm (SHARC) model is shown in Figure 1. It introduces a hierarchical structure in the GNN-based clustering. Figure 1(a), shows the training procedure using R audio recordings $r \in \{1, 2, .., R\}$ where r is the recording-id assigned to each recording in the dataset. It involves extracting short segment embeddings such as x-vectors $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_R\}$ from an Extended Time Delay Neural Network (ETDNN) [1] for all recordings where $\boldsymbol{X}_r \in \mathcal{R}^{N_r \times F}$, N_r is the number of x-vectors for recording r and F is the dimension of x-vectors. These are used to form graphs at different levels of hierarchy denoted as $G = \{G_1^0, G_2^0, ..., G_1^1, ..., G_R^{Mr}\}$ where G_r^m is a graph of recording r at level m and M_r is the maximum number of levels created for r. The nodes of the graphs are obtained from $\boldsymbol{\mathcal{X}}$, and edges are connected using k-nearest neighbors with weights coming from similarity matrices $\boldsymbol{\mathcal{S}^m} = \{\boldsymbol{S}_1^m, ..., \boldsymbol{S}_R^m\}$ for level m where, $\boldsymbol{S}_r^m \in \mathcal{R}^{N_r^m \times N_r^m}$, N_r^m is number of nodes at level m for recording r. The graphs are constructed at different clustering levels by merging the node features of each cluster and recomputing the similarity matrix. For training, a set of graphs G are fed to the GNN module in batches. The module comprises of GNN along with a feed forward network to predict edge weights $\hat{E_m} \in \mathcal{R}^{N_r^m \times k}$ of all nodes with their k-nearest neighbors. The loss is computed using E^q (true edge weights) and $\hat{E_m}$ (predicted edge weights).

Figure 1(b), shows the inference block diagram. For a test recording t, x-vectors $\boldsymbol{X}_t$ and $\boldsymbol{S}_t$ are extracted and a graph G_t^0 is constructed at level 0. Then, it is passed to the iterative clustering module using edge predictions from GNN module followed by merging nodes of same cluster and then, reconstructing graph for next level m. This process stops if the graph has no edges ($G^m = \{\phi\}$) or maximum allowed level M is attained. The algorithm (depicted in Algorithm 1) outputs cluster labels for the nodes at the top level, propagating down to the original embeddings.

3.1. Graph generation

In this step, a hierarchy of graphs, $G_r^m = (V^m, E_m)$, is created using $\boldsymbol{X}_r$ and $\boldsymbol{S}_r^m$ where $V^m = \{v_1, v_2, ..., v_n\}$ is the set of the nodes and E_m is the set of the edges. Each graph consists of node representations $H_r^m = \{h_1^{(m)}, h_2^{(m)}, ..., h_n^{(m)}\} \in \mathcal{R}^{F' X n}$ where $n = N_r^m$ is the number of nodes at level m. E_m is obtained using $\boldsymbol{S}_r^m \in [0, 1]$ considering k-nearest neighbors of each node in V^m. At level $m = 0$, we consider each embedding as individual cluster. Therefore, node representations are given as $H_r^0 = [\boldsymbol{X}_r; \boldsymbol{X}_r]$. For any level $m > 0$, the node representation is obtained by concatenating the identity feature and the average feature of the current cluster.

3.2. GNN scoring and clustering

The node representations H^m at each level m are passed to the GNN scoring function Φ. It predicts edge linkage probability (p_{ij}) which indicates presence of an edge $(v_i, v_j) \in V^m$ along with node density ($\hat{d}_i$) which measures how densely the node is connected with its neighbors.

After GNN scoring, the clustering is performed. At each level of hierarchy m, it creates a candidate edge set $\varepsilon(i)$, for the node v_i, with edge connection threshold p_τ, as given below.

$$\varepsilon(i) = \{j | (v_i, v_j) \in E_m, \quad \hat{d}_i \leq \hat{d}_j \quad \text{and} \quad p_{ij} \geq p_\tau\} \tag{2}$$

For any i, if $\varepsilon(i)$ is not empty, we pick $j = \text{argmax}_{j \in \varepsilon(i)} \hat{e}_{ij}$ and add (v_i, v_j) to E'_m where $\hat{e}_{ij}$ is the predicted edge weights, given as,

$$\hat{e}_{ij} = 2p_{ij} - 1 \in [-1, 1] \forall j \in N_i^k \tag{3}$$

Here, N_i^k are the k-nearest neighbors of node v_i. After a full pass over every node, E'_m forms a set of connected components C'_m, which serves as the designated clusters for the next level $(m + 1)$. The clustering stops when there are no connected components present in the graph.

Algorithm 1: SHARC Inference

Initialize: $M \leftarrow$ maximum no. of levels; $m \leftarrow 0$;
$k \leftarrow$ no. of nearest neighbors; $H^0 \leftarrow [\boldsymbol{X}; \boldsymbol{X}]$
while $m \leq M$ **do**

1. Graph generation: $G^m \leftarrow graph(H^m, \boldsymbol{S}, k)$
   If $(G^m \leftarrow \{\phi\}) : M = m; break$
2. GNN Scoring: $\hat{E}_m \leftarrow \Phi(G^m, H^m)$
3. $C'_m \leftarrow Clustering(\hat{E}_m)$
4. Aggregation: $H^{m+1} \leftarrow \Psi(H^m, C'_m)$
5. $m \leftarrow m + 1$

end
Output: Predicted $\hat{\boldsymbol{Y}} = \{\hat{y}_1, .., \hat{y}_N\}$ using $C'_{\{1:m-1\}}$

3.3. Feature aggregation

To obtain node representations for next level H^{m+1}, the connected components C'_m obtained from the clustering along with H^m are passed to an aggregation function Ψ . The function Ψ concatenates identity feature $\tilde{h}_i^{(m+1)}$ and average feature $\bar{h}_i^{(m+1)}$ of each cluster i to obtain $h_i{}^{(m+1)} = [\tilde{h}_i^{(m+1)}; \bar{h}_i^{(m+1)}]$. The identity feature of node i at level $m+1$ is the feature of the node which has highest node density at level m in the cluster i. The average feature is computed by taking average of all the identity features from previous level of that cluster, given as,

$$\tilde{h}_i^{(m+1)} = \tilde{h}_{z_i}^{(m)}; \qquad \bar{h}_i^{(m+1)} = \frac{1}{|c_i{}^{(m)}|} \sum_{j \in c_i{}^{(m)}} \tilde{h}_j^{(m)} \tag{4}$$

where z_i = argmax$_{j \in c_i{}^{(m)}} \hat{d}_j^{(m)}$.

3.4. GNN module architecture and training

GNN scoring function Φ is implemented as a learnable GNN module designed for supervised clustering. The module consists of one GraphSAGE [19] layer with $F' = 2048$ neurons. Each graph G_r^m, containing source and destination node pairs, is fed to the GNN module. It takes node representations H^m and their edge connections as input and generates latent representations denoted as $\hat{H}^{(m)} \in \mathcal{R}^{F' \times n}$, with n being the number of embeddings at a level m. The pair of embeddings are concatenated $[\hat{h}_i; \hat{h}_j]$ and passed to a three-layer fully connected feed-forward network with a size of $\{1024, 1024, 2\}$ followed by softmax activation to generate linkage probability p_{ij}. The predicted node density is computed as:

$$\hat{d}_i = \frac{1}{k} \sum_{j \in N_i^k} \hat{e}_{ij} \boldsymbol{S}_r(i, j) \tag{5}$$

The ground truth density d_i is obtained using ground truth edge weight $e_{ij}^q = 2q_{ij} - 1 \in E^q\{-1, 1\}^{N_r X k}$, where $q_{ij} = 1$ if nodes v_i and v_j belong to the same cluster, otherwise $q_{ij} = 0$. A node with higher density is a better representative of the cluster than a node with lower density.

Each node v_i has a cluster (speaker) label y_i in the training set, allowing the function to learn the clustering criterion from the data. The loss function for training is given as follows:

$$L = L_{conn} + L_{den} \tag{6}$$

Table 1. Choice of hyper-parameters for train, dev, eval split of AMI and Voxconverse datasets. The parameters k^* and p_τ^* are used in E2E-SHARC training.

Parameters	AMI			Voxconverse		
	Train	Dev	Eval	Train	Dev	Eval
k	60	60	60	60	30	30
p_τ	-	0.0	0.0	-	0.5	0.8
k^*	30	50	50	60	30	30
p_τ^*	-	0.0	0.0	-	0.9	0.8

where L_{conn} is the pairwise binary cross entropy loss based on linkage probability across all the possible edges in E accumulated across all levels and recordings in a batch. L_{den} represents mean squared error (MSE) loss between ground truth node density d_i and predicted node density $\hat{d}_i \; \forall i \in \{1, ..., |V|\}$, where $|V|$ is the cardinality of V.

3.5. E2E-SHARC

The SHARC model described in the previous section also allows the computation of gradients of the loss function w.r.t the input x-vector embeddings. The computation of these gradients enables the fine-tuning of the embedding extractor. We remove the classification layer from the 13-layer ETDNN model [20] and connect the 11^{th} affine layer output with the SHARC model input. This model is trained using 40-D mel-spectrogram feature vectors and similarity matrices as input. The details of ETDNN embedding extractor are described in Section 4.2. The training loss is the same as the SHARC model (Equation 6). This approach is referred as End-to-End Supervised HierArchical gRaph Clustering (E2E-SHARC).

4. EXPERIMENTS

4.1. Datasets

- **AMI train, dev and eval sets**: The AMI dataset [21] contains meeting recordings from four different sites (Edinburgh, Idiap, TNO, Brno). It comprises of training, development (dev) and evaluation (eval) sets consisting of 136, 18 and 16 recordings sampled at 16kHz, respectively. The number of speakers and the duration ranges of each recording from 3-5 and 20-60 mins, respectively.
- **Voxconverse dataset**: The dataset used for training Voxconverse model is simulated using Voxceleb 1 and 2 [22, 23] and Librispeech [24] using the recipe from [10]. We simulated 5000 mixtures containing 2-5 speakers with duration ranging from 150-440 s. This generates 1000 hrs of data with $6,023$ speakers. Voxconverse dev/eval sets are audio-visual diarization datasets [25] consisting of multi-speaker recordings extracted from YouTube videos. It is divided into a development (dev) set and an evaluation (eval) set consisting of 216 and 232 recordings respectively. The duration of a recording ranges from 22-1200 s. The number of speakers per recording varies from 1-21.

4.2. Baseline system

The baseline method is an x-vector-clustering based approach followed in [14, 26]. First, the recording is divided into 1.5 s short segments with 0.75 s shift. The 40-D mel-spectrogram features are computed from each segment which is passed to the ETDNN model [1] to extract 512-D x-vectors. The ETDNN model follows the BigDNN architecture described in [20] and is trained on the VoxCeleb1 [22] and VoxCeleb2 [23] datasets, for speaker identification task,

Table 2. DER (%) comparison on the AMI SDM and Voxconverse datasets with the baseline methods. OVP: overlap, COL: collar.

AMI SDM System	**with OVP + no COL**		**w/out OVP + COL**	
	Dev.	Eval.	Dev.	Eval.
x-vec + PLDA + AHC [26]	24.50	29.51	7.61	14.59
x-vec + PLDA + SC	19.8	22.29	4.1	5.76
x-vec + PLDA + SHARC	**19.71**	21.44	**3.91**	4.88
E2E-SHARC	20.59	**19.83**	5.15	**2.89**
– + VBx [27]	**19.35**	**19.82**	**3.46**	**2.73**
Voxconverse System				
x-vec + PLDA + AHC [26]	12.68	13.41	7.82	9.28
x-vec + PLDA + SC	10.78	14.02	6.52	9.92
x-vec + PLDA + SHARC	10.25	13.29	6.06	9.40
E2E-SHARC	**9.90**	**11.68**	**5.68**	**7.65**
– + VBx [27]	**8.29**	**9.67**	**3.94**	**5.51**

to discriminate among the $7,146$ speakers. The whitening transform, length normalization and recording level PCA are applied to the x-vectors as pre-processing steps to compute the PLDA similarity score matrix and perform clustering to generate speaker labels for each segment. For comparison, we have used two most popular clustering approaches - AHC [5] and SC [28]. To perform AHC, the PLDA is used directly. For SC, we convert the scores in $[0, 1]$ range by applying sigmoid with temperature parameter $\tau = 0.1$.

4.3. Training configuration

For training the SHARC model, we extract x-vectors with a window of duration 1.5 s with 0.75 s shift, from single speaker regions of the training set. The similarity score matrices, $\boldsymbol{S}^m$, are obtained using baseline PLDA models which are fed to the GNN module described in Section 3.4. The possible levels of each recording depend on the number of x-vectors (N_r) and the choice of k.

To train the end-to-end SHARC model, the weights of the x-vector model are initialized with the pre-trained ETDNN model while the SHARC model weights are initialized with the one obtained from SHARC training. The input to the model is 40-D mel-spectrogram computed from 1.5 s with 0.75 s shift. To prevent overfitting of the embedding extractor, the pre-trained x-vectors are added to the embedding extractor output before feeding to the GNN.

4.4. Choice of hyper-parameters

The SHARC model is trained with Stochastic Gradient Descent (SGD) optimizer with a learning rate lr=0.01 (for Voxconverse) and lr=0.001(for AMI) for 500 epochs. Similarly, the E2E-SHARC is also trained with an SGD optimizer. In this case, the learning rate is $1e$-06 for the ETDNN model and $1e$-03 for the SHARC model, trained for 20 epochs. The hyperparameters k, p_τ are selected based on the best performance on the dev set for the eval set and vice versa. The maximum number of levels M is initially set to 15, but the algorithm converges at $M \leq 3$. Table 1 shows the values of hyperparameters for AMI and Voxconverse.

5. RESULTS

The proposed approaches are evaluated using the diarization error rate (DER) metric [26]. In our work, we use ground truth speech regions for performance comparison. The DERs are computed for two cases. The first case considers overlaps and without collar regions, and the second case ignores overlaps and incorporates a tolerance collar of 0.25 s. Table 2 shows that the proposed SHARC model improves over the baseline systems, and the performance further improves with the E2E-SHARC model for both datasets. To incorporate temporal continuity, we applied a re-segmentation approach using Variational Bayes inference (VBx) [27] with the E2E-SHARC clustering labels as initialization, which further boosted the performance. As shown in Table 2, for the AMI SDM dataset, we obtain 15.6% and 52.6% relative improvements for the dev and eval set, respectively over the PLDA-SC baseline (best). Similarly, we achieve 39.6% and 44.4% relative improvements over the Voxconverse baseline (PLDA- SC) for the dev and eval set, respectively.

Table 3. DER (%, w/out overlap + with collar) comparison with state-of-the-art on AMI MDM (without TNO sets) and Voxconverse datasets.

AMI MDM System	Dev.	Eval.
x-vec(ResNet101)+AHC+VBx [29]	2.78	3.09
ECAPA-TDNN [30]	3.66	**3.01**
SelfSup-PLDA-PIC (+VBx) [14]	5.38 (**2.18**)	4.63 (3.27)
SHARC (+VBx)	3.58 (3.72)	**2.29 (2.11)**
Voxconverse System	Dev.	Eval.
Voxconverse challenge [25]	24.57	–
VBx BUT system [31]	4.36	–
Wang et. al. [32]	4.41	5.82
E2E-SHARC +VBx	**3.94**	**5.51**

Table 3 compares proposed approach performance with state-of-the-art systems. The widely reported beamformed AMI multi-distant microphone (MDM) dataset, without TNO recordings, is used for benchmarking. The beamformed recordings are obtained using [33].

The proposed SHARC model has the lowest DER for eval set compared to all previous SOTA approaches. For the Voxconverse dataset, we compare it with the challenge baseline and other published results. Here, the E2E-SHARC with VBx shows the best results compared to previously published results.

6. SUMMARY

We have proposed a supervised hierarchical clustering algorithm using graph neural networks for speaker diarization. The GNN module learns the edge linkages and node densities across all levels of hierarchy. The proposed approach enables the learnt GNN module to perform clustering hierarchically based on merging criteria which can handle a large number of speakers. The method is further extended to perform end-to-end diarization by jointly learning the embedding extractor and the GNN module. With challenging diarization datasets, we have illustrated the performance improvements obtained using the proposed approach.

Acknowledgements - The authors would like to thank Michael Free, Rohit Singh, Shakti Srivastava of British Telecom Research for their valuable inputs.

7. REFERENCES

[1] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *IEEE ICASSP*, 2019, pp. 5796–5800.

[2] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *IEEE ICASSP*, 2018, pp. 5329–5333.

[3] Sergey Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[4] Gregory Sell and Daniel Garcia-Romero, "Speaker diarization with PLDA i-vector scoring and unsupervised calibration," in *IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 413–417.

[5] William H. E. Day and Herbert Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, pp. 7–24, 1984.

[6] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno, "Speaker diarization with LSTM," in *IEEE ICASSP*, 2018, pp. 5239–5243.

[7] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[8] Wei Zhang, Deli Zhao, and Xiaogang Wang, "Agglomerative clustering via maximum incremental path integral," *Pattern Recognition*, vol. 46, no. 11, pp. 3056–3065, 2013.

[9] Prachi Singh and Sriram Ganapathy, "Self-supervised representation learning with path integral clustering for speaker diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1639–1649, 2021.

[10] Yusuke Fujita et al., "End-to-End Neural Speaker Diarization with Self-attention," in *ASRU*, 2019, pp. 296–303.

[11] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Kenji Nagamatsu, "End-to-End Speaker Diarization for an Unknown Number of Speakers with Encoder-Decoder Based Attractors," in *INTERSPEECH*, 2020.

[12] Shota Horiguchi, Shinji Watanabe, Paola García, Yawen Xue, Yuki Takashima, and Yohei Kawaguchi, "Towards neural diarization for unlimited numbers of speakers using global and local attractors," in *IEEE ASRU*, 2021, pp. 98–105.

[13] Qingjian Lin, Ruiqing Yin, Ming Li, Hervé Bredin, and Claude Barras, "LSTM Based Similarity Measurement with Spectral Clustering for Speaker Diarization," in *Proc. INTERSPEECH*, 2019, pp. 366–370.

[14] Prachi Singh and Sriram Ganapathy, "Self-supervised metric learning with graph clustering for speaker diarization," in *IEEE ASRU*, 2021, pp. 90–97.

[15] Yifan Xing, Tong He, Tianjun Xiao, Yongxin Wang, Yuanjun Xiong, Wei Xia, David Wipf, Zheng Zhang, and Stefano Soatto, "Learning hierarchical graph neural networks for image clustering," in *Proc. IEEE ICCV*, 2021, pp. 3467–3477.

[16] Jixuan Wang, Xiong Xiao, Jian Wu, Ranjani Ramamurthy, Frank Rudzicz, and Michael Brudno, "Speaker diarization with session-level speaker embedding refinement using graph neural networks," in *IEEE ICASSP*, 2020, pp. 7109–7113.

[17] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] Fuchuan Tong, Siqi Zheng, Min Zhang, Yafeng Chen, Hongbin Suo, Qingyang Hong, and Lin Li, "Graph convolutional network based semi-supervised learning on multi-speaker meeting data," in *IEEE ICASSP*, 2022, pp. 6622–6626.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[20] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[21] I McCowan, J Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, et al., "The AMI meeting corpus," in *International Conference on Methods and Techniques in Behavioral Research*, 2005, pp. 137–140.

[22] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: A large-scale speaker identification dataset," *Proc. of INTERSPEECH*, pp. 2616–2620, 2017.

[23] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. of INTERSPEECH*, 2018, pp. 1086–1090.

[24] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *IEEE ICASSP*, 2015, pp. 5206–5210.

[25] Joon Son Chung, Jaesung Huh, Arsha Nagrani, Triantafyllos Afouras, and Andrew Zisserman, "Spot the Conversation: Speaker Diarisation in the Wild," in *Proc. INTERSPEECH 2020*, 2020, pp. 299–303.

[26] Neville Ryant, Prachi Singh, Venkat Krishnamohan, Rajat Varma, Kenneth Church, Christopher Cieri, Jun Du, Sriram Ganapathy, and Mark Liberman, "The Third DIHARD Diarization Challenge," in *Proc. INTERSPEECH*, 2021, pp. 3570–3574.

[27] Federico Landini et al., "BUT system for the second DIHARD speech diarization challenge," in *IEEE ICASSP*, 2020, pp. 6529–6533.

[28] Andrew Ng, Michael Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

[29] Federico Landini et al., "Bayesian HMM clustering of x-vector sequences (VBx) in speaker diarization: theory, implementation and analysis on standard tasks," *arXiv preprint arXiv:2012.14952*, 2020.

[30] Nauman Dawalatabad et al., "ECAPA-TDNN embeddings for speaker diarization," *arXiv preprint arXiv:2104.01466*, 2021.

[31] Federico Landini, Ondřej Glembek, Pavel Matějka, Johan Rohdin, Lukáš Burget, Mireia Diez, and Anna Silnova, "Analysis of the BUT diarization system for voxconverse challenge," in *IEEE ICASSP*, 2021, pp. 5819–5823.

[32] Weiqing Wang et al., "Similarity measurement of segment-level speaker embeddings in speaker diarization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2645–2658, 2022.

[33] X. Anguera, C. Wooters, and J. Hernando, "Acoustic beamforming for speaker diarization of meetings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2011–2021, 2007.