WESPEAKER: A RESEARCH AND PRODUCTION ORIENTED SPEAKER EMBEDDING LEARNING TOOLKIT

Hongji Wang[1,2,7], *Chengdong Liang*[3,4,7], *Shuai Wang*[1,7,*], *Zhengyang Chen*[1], *Binbin Zhang*[4,7], *Xu Xiang*[5], *Yanlei Deng*[6], *Yanmin Qian*[1]

[1]X-LANCE Lab, Shanghai Jiao Tong University, Shanghai, China
[2]Tencent Ethereal Audio Lab, Tencent Corporation, Shenzhen, China
[3]School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China
[4]Horizon Robotics, Beijing, China,[5]AISpeech Ltd, Suzhou, China,[6]NVIDIA, Santa Clara, USA
[7]WeNet Open Source Community

ABSTRACT

Speaker modeling is essential for many related tasks, such as speaker recognition and speaker diarization. The dominant modeling approach is fixed-dimensional vector representation, i.e., speaker embedding. This paper introduces a research and production oriented speaker embedding learning toolkit, Wespeaker. Wespeaker contains the implementation of scalable data management, state-of-the-art speaker embedding models, loss functions, and scoring backends, with highly competitive results achieved by structured recipes which were adopted in the winning systems in several speaker verification challenges. The application to other downstream tasks such as speaker diarization is also exhibited in the related recipe. Moreover, CPU- and GPU-compatible deployment codes are integrated for production-oriented development. The toolkit is publicly available at `https://github.com/wenet-e2e/wespeaker`.

***Index Terms*—** Wespeaker, Speaker embedding, Speaker verification, Speaker diarization

1. INTRODUCTION

Deep speaker embeddings [1, 2, 3, 4] are the de facto standard speaker identity representation for many related tasks. For speaker recognition, they are fed to scoring back-ends such as cosine similarity or probabilistic linear discriminant analysis (PLDA) for the following acceptance decision making. A similar application can be found in speaker diarization, where the scores obtained are used for further clustering. Besides the tasks that focus on speaker modeling, deep speaker embeddings are also utilized in other speech processing tasks, such as the speaker adaptation in speech recognition, speaker modeling in text-to-speech and voice conversion, etc.

Researchers in the intelligent speech processing community have been quite active in the open-source field. The early general speech processing toolkits, such as HTK [5] and Kaldi [6], had equipped many researchers and industrial productions before deep learning toolkits such as Pytorch [7] and Tensorflow [8], while the recently thriving Pytorch based SpeechBrain [9], Espnet [10], are much more friendly for new researchers and enables fast prototyping. Unlike the mentioned general-purpose speech processing toolkits, Wenet [11, 12] focuses on end-to-end speech recognition and is designed to bridge the gap between the research and deployment.

* Shuai Wang is the corresponding author.

Competitions such as the VoxSRC series [13, 14, 15] and CNSRC 2022 significantly promote the related dataset while accordingly inspiring researchers' creativity and engineering ability, leading to new SOTA results. However, there is usually a clear gap between the results reported in the research papers and competition system descriptions. The main reason might be that tricks and dedicated engineering efforts are usually neglected for the former. Another problem is that current speaker-related open-source implementations are research focused, without the support for potential migration for the production environment.

To this end, we would like to design a speaker embedding learning toolkit that provides clean and well-structured codes for learning high-quality embeddings, with good portability to the production scenarios. Following Wenet, we propose an open-source Wespeaker which focuses on deep speaker embedding learning as another project of this "we"-series. The key features/advantages of the Wespeaker toolkit are as follows,

- **Competitive results**: Compared with other open-source implementations [9, 16], we achieve very competitive performance in all the recipes, including the VoxCeleb, CNCeleb, and VoxConverse. Many tricks used in the winning systems of the related competitions are re-implemented in Wespeaker to boost the system's performance. We hope Wespeaker can provide the researchers with a competitive starting point for their algorithm innovation.
- **Light-weight**: Wespeaker is designed specifically for deep speaker embedding learning with clean and simple codes. It is purely built upon PyTorch and its ecosystem, and has no dependencies on Kaldi [6].
- **Unified IO (UIO)**: A unified IO system similar to the one used in Wenet [12] is introduced, providing a unified interface that can elastically support training with a few hours to millions of hours of data.
- **On-the-fly feature preparation**: Unlike the traditional feature preparation procedure, which performs utterance segmentation, data augmentation and feature extraction in an offline manner, Wespeaker performs all the above steps in an on-the-fly manner. Different augmentation methods, including signal-level ones such as noise corruption, reverberation, resampling, speed perturbation, and feature-level SpecAug [17], are supported.

- **Distributed training**: Wespeaker supports distributed training to speed up, where "DistributedDataParallel" in Pytorch is adopted for multi-node multi-GPU scalability.
- **Production ready**: All models in Wespeaker can be easily exported by torch Just In Time (JIT) or as the ONNX format, which can be easily adopted in the deployment environment. Sample deployment codes are also provided.

2. WESPEAKER

2.1. Deep speaker embedding learning

For a standard deep speaker embedding learning system, the input is frame-level features (e.g., Fbank) and the expected output is segment-level embeddings. Such systems usually consist of several frame-level layers to process the input features, followed by a pooling layer to aggregate the encoded frame-level information into segment-level representations, and then several (commonly one or two) segment-level transform layers that map these representations to the correct speaker labels. Moreover, a class-based or metric-based loss function is adopted to provide a speaker-discriminative supervision signal.

2.2. Overall structure

Figure 1 shows the overall structure and pipeline of Wespeaker. A standard procedure contains data preparation on the disk, online feature preparation, and model training. After the converged model is obtained, it can be easily exported to a run-time format and ready for further deployment. The extracted speaker embeddings can then be applied to downstream tasks, such as speaker verification and diarization.

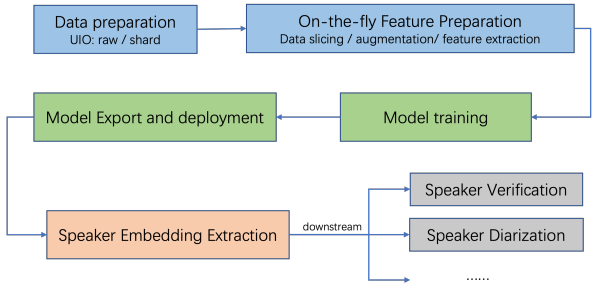

Fig. 1. The overall structure of Wespeaker

2.3. Data management

2.3.1. *Unified IO*

Production-scale corpus usually contains tens of thousands of hours of speech, which are comprised of massive small files. To avoid the possible consequent out-of-memory (OOM) and slow-training problems, we introduce the unified IO (UIO) mechanism in Wenet[1] to the data management in Wespeaker. This mechanism was inspired by the TFRecord format used in Tensorflow [8] and AIStore [18], which packs each set of small files into a bigger shard via the GNU tar. As shown in Figure 2, for the large dataset, on-the-fly decompression will be performed to sequentially read the shard files into the memory during the training stage. On the other hand, for the small dataset, Wespeaker supports the traditional data loading functions to load the raw files from the disk directly.

[1] https://wenet.org.cn/wenet/UIO.html

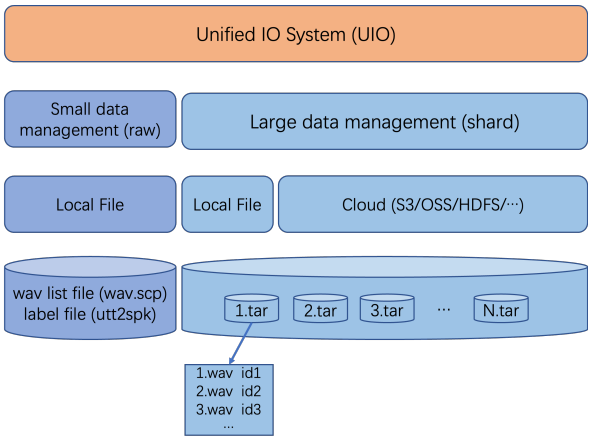

Fig. 2. Unified IO in Wespeaker

2.3.2. *On-the-fly feature preparation*

Traditional feature preparation for speaker embedding learning is usually done offline. A typical offline procedure could comprise resampling, data augmentation, data slicing[2] and feature extraction. The offline feature preparation generates the final training examples and saves them on the disk, which will remain unchanged during the whole training process[3]. Wespeaker loads the original wave data and performs all these steps in an on-the-fly manner, which has two main advantages: 1) There is no need to save the augmented wave files and processed features, which significantly saves the disk cost. 2) Online augmentation makes it possible for the model to see different training examples at different epochs, this uncertainty and randomness improve the robustness of the resultant model.

Figure 3 presents the pipeline of online feature preparation in Wespeaker, which includes the following modules:

- Local shuffle: Construct a local buffer and shuffle it randomly each time, contributing to different training batches at different epochs.
- Spk2id: Map speaker name into speaker id (from 0 to N-1, where N is the total speaker number of the training set).
- Resample: Resample the training data into the targeted sample rate.
- Speed perturb: Change the speed of the training data with a certain probability.
- Random chunk: Chunk the training data into the same length randomly. Padding is applied on the short ones.
- Noise/Reverb: Add noise or reverberation augmentation.
- Compute Fbank: Extract Fbank feature from raw PCM data.
- CMVN: Apply cepstral mean and variance normalization per utterance.
- SpecAug[4]: Apply Spec-Augmentation [17] on the feature.
- Batch: Organize the training data into fixed-size batches.

[2] Data slicing means cutting each utterance into a fixed length, which eases the data preparation and speeds up the GPU training. For text-independent speaker modeling, we assume this context information corruption has a limited impact on the modeling accuracy.

[3] We name this "feat" data type and also support it in Wespeaker.

[4] SpecAug is commonly not compatible with other augmentation types and thus set to False by default.

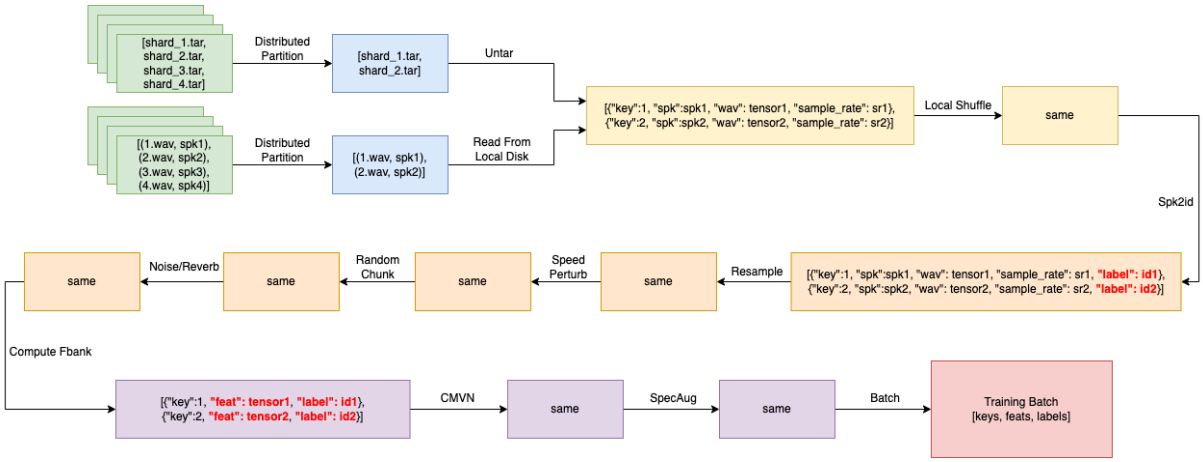

Fig. 3. The pipeline of online feature preparation in Wespeaker. "same" means that no new attributes are added and only change a specific attribute value for each sample (except local shuffle), compared with the last node.

2.4. SOTA Model Implementation

Wespeaker currently supports the following models,

- TDNN based x-vector, this is a milestone work that leads the following deep speaker embedding era.
- ResNet based r-vector and its deeper version, this is the winning system of VoxSRC 2019 [3] and CNSRC 2022 [19].
- ECAPA-TDNN, a modified version of TDNN, this is the winning system of VoxSRC 2020 [4].

Pooling functions aggregate frame-level features into segment-level representations, where Wespeaker supports the statistics-based and attention-based ones. **Loss functions** are critical to current deep speaker embedding learning. We support the standard softmax cross-entropy loss and different margin-based variants [20, 21], such as A-softmax [22, 23], AM-softmax [24] and AAM-softmax [25].

2.5. Training strategies

2.5.1. *Learning Rate and Margin Schedule*

In Wespeaker implementation, the learning rate schedule is the result of two functions working together. Here, we denote the variation function of the learning rate with respect to time as $lr(t)$ and it can be represented by the product of warmup function $g(t)$ and exponential descent function $h(t)$: $lr(t) = g(t)h(t)$. The specific expressions of $g(t)$ and $h(t)$ are:

$$g(t) = \begin{cases} \frac{t}{T_{warm}}, & t < T_{warm}. \\ 1, & T_{warm} \leq t < T. \end{cases} \tag{1}$$

$$h(t) = \eta_0 \cdot e^{\left(\frac{t}{T} ln(\frac{\eta_T}{\eta_0})\right)} \tag{2}$$

where t is the current training iteration, T_{warm} is the warm-up iteration, T is the total iteration, η_0 is the initial learning rate and η_T is the final learning rate.

The scheduler for the margin in loss is a three-stage function $m(t)$:

$$m(t) = \begin{cases} 0, & t < T_1. \\ f(t), & T_1 \leq t < T_2. \\ M, & T_2 \leq t < T. \end{cases} \tag{3}$$

where $0 \leq T_1 \leq T_2 \leq T$, M is the final margin in loss and $f(t)$ is a linear growth function or a logarithmic growth function from 0 to M.

2.5.2. *Large Margin Fine-tuning*

The large margin fine-tuning strategy was first proposed in [26] and widely used in speaker verification challenge systems [19, 27, 28] to further enhance the system's performance. This strategy is performed as an additional fine-tuning stage based on a well-trained speaker verification model. In this stage, the model will be trained with a larger margin and longer training segments relative to the normal training stage. For Wespeaker implementation, we use AAM loss with a margin of 0.5 and 6s training segments.

2.6. Back-end Support

For the dominant deep speaker embeddings supervised by large-margin softmax losses, the simple cosine similarity can serve as a good scoring back-end. Before the era of large-margin embeddings, parametric back-ends such as probabilistic linear discriminant analysis (PLDA) are more widely used. Wespeaker implements both scoring back-ends[5], while an additional score normalization function [29] is also provided to calibrate the speaker verification scores.

2.7. Deployment

For the models trained in Wespeaker, we can easily export them to "tensorrt" or "onnx" format, which can be deployed on the Triton Inference Server. Detailed information including the instructions and performance can be found at `https://github.com/wenet-e2e/wespeaker/tree/master/runtime/server/x86_gpu`. Furthermore, since Wespeaker is designed as a general speaker embedding learner, we also provide the python bindings and deploy it via standard "pip" packaging, which allows users to trivially use the pre-trained models for down-stream tasks[6].

[5]The two-covariance version of PLDA is supported currently, more variants will be added in the future

[6]`https://github.com/wenet-e2e/wespeaker/tree/master/runtime/binding/python`, a toy demo on speaker verification can be found at `https://huggingface.co/spaces/wenet/wespeaker_demo`

3. EXPERIMENTS AND RECIPES

As described in the above sections, deep speaker embeddings could be applied to different downstream tasks, whereas this paper focuses on speaker verification and speaker diarization.

3.1. Basic setups for speaker embedding learning

For the training setups of all speaker models in the following sections, we adopted the shard UIO method and applied the same online data augmentation in the training process. The audios from the MUSAN dataset [30] are used as additive noises, while the simulated room impulse responses (RIRs)[7] are used for the reverberation. For each utterance in the training set, we apply additive-noise or reverberation augmentation (not both at the same time) with a probability of 0.6. For speed perturbation, we randomly change the speed of an utterance with a ratio of 0.9 or 1.1, and the augmented audios will be treated as from new speakers due to the pitch shift after the augmentation. Moreover, the ratio of speeds 0.9, 1.0 and 1.1 is set as 1:1:1. The acoustic features are 80-dimensional log Mel-filter banks (Fbank) with a 10ms frameshift and a 25ms frame window. All training data are chunked into 200 frames and CMN (without CVN) is also applied. Note that SpecAug is not used in all experiments.

3.2. Speaker Verification

For the speaker verification task, recipes are constructed based on the VoxCeleb and CNCeleb datasets, which are very popular and used by many researchers, thanks to the promotion by related competitions. All the results exhibited here are obtained after large margin fine-tuning, with cosine scoring and AS-Norm applied.[8].

3.2.1. VoxCeleb

VoxCeleb dataset [31] was released by Oxford and has become one of the most popular text-independent speaker recognition datasets. Following the segmentation of the VoxSRC challenge, we only use the VoxCeleb2 dev as the training set, which contains more than one million audios from 5994 speakers.

Table 1. Results achieved using different architectures on the VoxCeleb dataset, "dev" of part 2 is used as the training set

Architecture	voxceleb1_O		voxceleb1_E		voxceleb1_H	
	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
TDNN	1.590	0.166	1.641	0.170	2.726	0.248
ECAPA-TDNN ([4])	0.870	0.107	1.120	0.132	2.120	0.210
ECAPA-TDNN	0.728	0.099	0.929	0.100	1.721	0.169
ResNet34 ([3])	1.31	0.154	1.38	0.163	2.50	0.233
ResNet34	0.723	0.069	0.867	0.097	1.532	0.146
ResNet221	0.505	0.045	0.676	0.067	1.213	0.111
ResNet293	**0.447**	**0.043**	**0.657**	**0.066**	**1.183**	**0.111**

The results in Table 1 show that our implementation achieves very competitive numbers compared with the original ones in the literature. Scaling the ResNet deeper can further boost the performance significantly.

[7] https://www.openslr.org/28

[8] Due to the limit of paper length, we only list the main results in the following experiments. More detailed results including ablation study, PLDA results, etc. can be found in the corresponding recipes online.

3.2.2. CNCeleb

For the CNCeleb recipe, we combine the 1996 speakers from CNCeleb2 and 797 speakers from the CNCeleb1 dev set as the training set, and evaluate on the CNCeleb1 test set. Although the collection procedure of the CNCeleb dataset [32] is similar to the one of VoxCeleb, many recordings are shorter than 2 seconds in this dataset. Therefore, in the data preparation, we first concatenate the short audios from the same genre and same speaker to construct audios longer than 5 seconds.

Table 2. Results on the CNCeleb evaluation set

Architecture	EER(%)	minDCF
TDNN	8.960	0.446
ECAPA-TDNN	7.395	0.372
ResNet34([16])	9.141	0.463
ResNet34	6.492	0.354
ResNet221	**5.655**	**0.330**

The results obtained using different backbones are exhibited in Table 2. Unlike the VoxCeleb evaluation protocol, CNCeleb assumes each speaker is enrolled with multiple sessions. Embeddings for all enrollment sessions for each speaker are extracted and averaged to obtain the final enrollment embedding, which brings considerable performance improvement in our experiments.

3.3. Speaker Diarization

VoxConverse dataset [33] was released for the diarization track in VoxSRC 2020, which is a "in the wild" dataset collected from the Youtube Videos. This recipe shows how to leverage a pre-trained speaker model for the speaker diarization task. The pre-trained ResNet34 model is used for speaker embedding extraction and spectral clustering is implemented specifically for this task. As illustrated in Table 3, we achieve strong results on the VoxConverse dev dataset, using oracle speech activity detection (SAD) annotations or system SAD results from Silero-VAD [34] pre-trained model, proving the effectiveness of deep speaker embedding learning in Wespeaker.

Table 3. Results on the VoxConverse dev set

System	MISS(%)	FA(%)	SC(%)	DER(%)
[33] (system SAD)	2.4	2.3	3.0	7.7
Wespeaker (system SAD)	4.4	0.6	2.1	7.1
Wespeaker (oracle SAD)	2.3	0.0	1.9	4.2

4. CONCLUSION AND FUTURE WORK

In this paper, we introduced Wespeaker, a research and production oriented speaker embedding learning toolkit. Wespeaker has a lightweight code base and focuses on high-quality speaker embedding learning, achieving very competitive results on several datasets. Despite the friendliness for researchers, CPU-and GPU- compatible deployment codes are also integrated to bridge the gap between the research and production systems.

For the next release, we will focus on the following key points: 1) Self-supervised learning (SSL) for speaker embedding learning. 2) Small footprint solutions for resource-limited scenarios. 3) Continually adding SOTA speaker models (network architectures and scoring back-ends) and optimizing the training strategies.

5. REFERENCES

[1] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[2] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[3] 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.

[4] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *arXiv preprint arXiv:2005.07143*, 2020.

[5] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al., "The htk book," *Cambridge university engineering department*, vol. 3, no. 175, pp. 12, 2002.

[6] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number CONF.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[8] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[9] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, et al., "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[10] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson-Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al., "Espnet: End-to-end speech processing toolkit," *Proc. Interspeech 2018*, pp. 2207–2211, 2018.

[11] Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei, "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," *arXiv preprint arXiv:2102.01547*, 2021.

[12] Binbin Zhang, Di Wu, Zhendong Peng, Xingchen Song, Zhuoyuan Yao, Hang Lv, Lei Xie, Chao Yang, Fuping Pan, and Jianwei Niu, "Wenet 2.0: More productive end-to-end speech recognition toolkit," *arXiv preprint arXiv:2203.15455*, 2022.

[13] Joon Son Chung, Arsha Nagrani, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "Voxsrc 2019: The first voxceleb speaker recognition challenge," *arXiv preprint arXiv:1912.02522*, 2019.

[14] Arsha Nagrani, Joon Son Chung, Jaesung Huh, Andrew Brown, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "Voxsrc 2020: The second voxceleb speaker recognition challenge," *arXiv preprint arXiv:2012.06867*, 2020.

[15] Andrew Brown, Jaesung Huh, Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxsrc 2021: The third voxceleb speaker recognition challenge," *arXiv preprint arXiv:2201.04583*, 2022.

[16] Fuchuan Tong, Miao Zhao, Jianfeng Zhou, Hao Lu, Zheng Li, Lin Li, and Qingyang Hong, "Asv-subtools: Open source toolkit for automatic speaker verification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6184–6188.

[17] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[18] Alex Aizman, Gavin Maltby, and Thomas Breuel, "High performance i/o for large scale deep learning," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5965–5967.

[19] Zhengyang Chen, Bei Liu, Bing Han, Leying Zhang, and Yanmin Qian, "The sjtu x-lance lab system for cnsrc 2022," *arXiv preprint arXiv:2206.11699*, 2022.

[20] Mahdi Hajibabaei and Dengxin Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[21] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 1652–1656.

[22] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[23] Zili Huang, Shuai Wang, and Kai Yu, "Angular softmax for short-duration text-independent speaker verification," *Proc. Interspeech 2018*, pp. 3623–3627, 2018.

[24] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[25] Jiankang Deng, Jia Guo, Xue Niannan, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "The idlab voxsrc-20 submission: Large margin fine-tuning and quality-aware score calibration in dnn based speaker verification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5814–5818.

[27] Miao Zhao, Yufeng Ma, Min Liu, and Minqiang Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[28] Rostislav Makarov, Nikita Torgashov, Alexander Alenin, Ivan Yakovlev, and Anton Okhotnikov, "Id r&d system description to voxceleb speaker recognition challenge 2022," 2022.

[29] Pavel Matejka, Ondrej Novotnỳ, Oldrich Plchot, Lukas Burget, Mireia Diez Sánchez, and Jan Cernockỳ, "Analysis of score normalization in multilingual speaker recognition.," in *Interspeech*, 2017, pp. 1567–1571.

[30] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[31] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, pp. 101027, 2020.

[32] Lantian Li, Ruiqi Liu, Jiawen Kang, Yue Fan, Hao Cui, Yunqi Cai, Ravichander Vipperla, Thomas Fang Zheng, and Dong Wang, "Cn-celeb: multi-genre speaker recognition," *Speech Communication*, vol. 137, pp. 77–91, 2022.

[33] Joon Son Chung, Jaesung Huh, Arsha Nagrani, Triantafyllos Afouras, and Andrew Zisserman, "Spot the conversation: speaker diarisation in the wild," *arXiv preprint arXiv:2007.01216*, 2020.

[34] Silero Team, "Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier," `https://github.com/snakers4/silero-vad`, 2021.