Speech Communication 162 (2024) 103104

Contents lists available at ScienceDirect

Speech Communication

journal homepage: www.elsevier.com/locate/specom

Advancing speaker embedding learning: Wespeaker toolkit for research and production

Shuai Wang [a,b,g,*], Zhengyang Chen [c], Bing Han [c], Hongji Wang [d,g], Chengdong Liang [g], Binbin Zhang [g], Xu Xiang [c], Wen Ding [e], Johan Rohdin [f], Anna Silnova [f], Yanmin Qian [c], Haizhou Li [b,a]

[a] Shenzhen Institute of Big Data, Shenzhen, Guangdong, China
[b] School of Data Science, The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), Guangdong, China
[c] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[d] Tencent Ethereal Audio Lab, Tencent Corporation, Shenzhen, China
[e] NVIDIA, Shanghai, China
[f] Brno University of Technology, Faculty of Information Technology, IT4I Centre of Excellence, Czechia
[g] WeNet Open Source Community, Suzhou, Jiangsu, China

ARTICLE INFO

Dataset link: https://github.com/wenet-e2e/wespeaker

Keywords:
Wespeaker
Speaker embedding learning
SSL
Open-source

ABSTRACT

Speaker modeling plays a crucial role in various tasks, and fixed-dimensional vector representations, known as speaker embeddings, are the predominant modeling approach. These embeddings are typically evaluated within the framework of speaker verification, yet their utility extends to a broad scope of related tasks including speaker diarization, speech synthesis, voice conversion, and target speaker extraction. This paper presents Wespeaker, a user-friendly toolkit designed for both research and production purposes, dedicated to the learning of speaker embeddings. Wespeaker offers scalable data management, state-of-the-art speaker embedding models, and self-supervised learning training schemes with the potential to leverage large-scale unlabeled real-world data. The toolkit incorporates structured recipes that have been successfully adopted in winning systems across various speaker verification challenges, ensuring highly competitive results. For production-oriented development, Wespeaker integrates CPU- and GPU-compatible deployment and runtime codes, supporting mainstream platforms such as Windows, Linux, Mac and on-device chips such as horizon X3'PI. Wespeaker also provides off-the-shelf high-quality speaker embeddings by providing various pretrained models, which can be effortlessly applied to different tasks that require speaker modeling. The toolkit is publicly available at https://github.com/wenet-e2e/wespeaker.

1. Introduction

Speech contains rich information, and speaker information is one of its most crucial aspects. Speaker information modeling is essential for speaker recognition tasks, which are widely used in security access systems, telephone banking, and other domains to enhance system security and improve user experience. Moreover, speaker information modeling is extensively utilized in various areas such as speaker diarization, speech synthesis, voice conversion, and target speaker extraction. Among various modeling strategies, deep speaker embeddings (Variani et al., 2014; Snyder et al., 2018; Zeinali et al., 2019; Desplanques et al., 2020) have become the standard representation for speaker identity in these related tasks. In speaker recognition, these embeddings are inputted into scoring back-ends like cosine similarity or probabilistic linear discriminant analysis (PLDA) to make acceptance decisions. A similar application is found in speaker diarization, where the obtained scores are used for further clustering. In tasks like target speaker extraction (Wang et al., 2018b; Zmolikova et al., 2023) and speech generation (Jia et al., 2018; Lu et al., 2019; Cooper et al., 2020; Cai et al., 2020; Cho et al., 2022; Zhao et al., 2023), deep speaker embeddings serve as auxiliary input to indicate the speaker's identity. Most existing deep speaker embedding learning systems are based on supervised training. The mainstream approach is to train a neural network with speaker classification as the target, thereby extracting the corresponding deep speaker embeddings from a specific layer (Snyder

* Corresponding author.
E-mail addresses: wangshuai@cuhk.edu.cn (S. Wang), zhengyang.chen@sjtu.edu.cn (Z. Chen), hanbing97@sjtu.edu.cn (B. Han), jijijiang77@sjtu.edu.cn (H. Wang), chengdong01.liang@horizon.cc (C. Liang), binbin.zhang@horizon.cc (B. Zhang), chinoiserie@sjtu.edu.cn (X. Xiang), wend@nvidia.com (W. Ding), rohdin@fit.vut.cz (J. Rohdin), isilnova@fit.vut.cz (A. Silnova), yanminqian@sjtu.edu.cn (Y. Qian), haizhouli@cuhk.edu.cn (H. Li).

https://doi.org/10.1016/j.specom.2024.103104
Received 4 February 2024; Received in revised form 30 April 2024; Accepted 10 July 2024
Available online 14 July 2024

Table 1
Existing open-source toolkits which support deep speaker embedding learning.

Toolkit	Speaker-task specific	Backbone Support	SSL Support	Deployment Support	Back-ends
Kaldi (Povey et al., 2011)	No	TDNN	No	No	Cosine, PLDA
SpeechBrain (Ravanelli et al., 2021)	No	TDNN, ECAPA-TDNN, ResNet	No	No	Cosine
NeMo	No	TitaNet	No	Yes	Cosine
VoxCeleb_Trainer (Nagrani et al., 2020b)	Yes	RawNet, ResNet34	No	No	Cosine
ASV-Subtools (Tong et al., 2021)	Yes	TDNN, ECAPA-TDNN, ResNet, Conformer	No	No	Cosine, PLDA
3D-Speaker	Yes	CAM++, ERes2Net,ECAPA-TDNN	RDINO	No	Cosine
Wespeaker	Yes	TDNN, ECAPA-TDNN ResNet, ERes2Net CAM++, RepVGG	SimCLR MoCo DINO	Yes	Cosine PLDA TPSDA

et al., 2018; Zeinali et al., 2019). Nevertheless, collecting large-scale labeled data presents a challenge and may encroach upon personal privacy, particularly considering that the quantity of data (number of speakers) significantly influences the quality of learned speaker embeddings. To leverage the massive data without speaker labels, self-supervised learning algorithms such as SimCLR (Chen et al., 2020), MoCo (He et al., 2020), and DINO (Caron et al., 2021), which have been widely applied in other fields, have also been introduced into the field of speaker modeling (Cai et al., 2021; Han et al., 2022; Cho et al., 2022; Zhang and Yu, 2022; Chen et al., 2023a), showing promising potential.

On the other hand, researchers are also designing and releasing various datasets. These datasets primarily fall into two categories. The first type is deliberately recorded (Larcher et al., 2012; Panayotov et al., 2015; Zheng et al., 2023), while the second type is semi-automatically generated by collecting videos from the internet, utilizing a labeling pipeline incorporating facial recognition, voice tracking, and manual cleaning. The latter type usually exhibits better generalization ability since they are collected from real scenarios. Open-source datasets like VoxCeleb (Nagrani et al., 2020b) and CNCeleb (Li et al., 2022) have enabled researchers to assess and compare system performance, both in supervised and unsupervised methods. Despite the availability of these datasets, the literature often reveals inconsistencies in directly comparing results on the same dataset, even when researchers claim to employ similar model structures. Concurrently, competitions like the VoxSRC series[1] (Chung et al., 2019; Nagrani et al., 2020a; Brown et al., 2022) and CNSRC 2022[2] play a significant role in promoting related datasets. These competitions stimulate researchers' creativity and engineering prowess, resulting in new state-of-the-art (SOTA) results. However, a noticeable disparity typically exists between the results reported in research papers and the descriptions of systems used in competitions. This discrepancy may arise because the former often overlooks the incorporation of certain techniques and dedicated engineering efforts utilized in the latter.

To some extent, the aforementioned challenges hinder direct comparisons of work from different groups and have also spurred the development of open-source tools.

The speech processing community has demonstrated high activity in the open-source domain. Initially, foundational speech processing toolkits like HTK (Young et al., 2002) and Kaldi (Povey et al., 2011) served as crucial resources for both researchers and industrial applications. These toolkits were instrumental before the advent of deep learning frameworks such as PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2016). More recently, PyTorch-based toolkits like SpeechBrain Ravanelli et al. (2021) and Espnet (Watanabe et al., 2018) have emerged, offering a more accessible interface for novice researchers and facilitating rapid prototyping.

In contrast to the aforementioned general-purpose speech processing toolkits, Wenet (Yao et al., 2021; Zhang et al., 2022) is specifically designed for end-to-end speech recognition. Its primary objective is to bridge the gap between research and deployment, focusing on enhancing the seamless integration of speech recognition models from experimental phases to real-world applications.

As part of the Wenet open-source projects, we released the initial version of Wespeaker in (Wang et al., 2023b). Since the first release, we have made significant updates, including the addition of different types of self-supervised algorithms, more recipes, and codes specifically designed for production. This paper aims to provide a comprehensive review of the design and capabilities of Wespeaker, as well as include the latest updates and performance comparisons across various toolkits, model architectures, and datasets. A collection and comparison of different open-source toolkits that enables speaker embedding learning is listed in Table 1.

The key features/advantages of the Wespeaker toolkit are as follows,

- **Competitive Results**: Compared with other open-source implementations (Ravanelli et al., 2021; Tong et al., 2021), we achieve very competitive performance in all the recipes, including the VoxCeleb, CNCeleb, and NIST SRE. 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 (Povey et al., 2011).
- **Unified IO (UIO)**: A unified IO system similar to the one used in Wenet (Zhang et al., 2022) 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 (Park et al., 2019), are supported.
- **SSL Support**: To leverage unlabeled data, Wespeaker presents the implementation of several popular self-supervised learning algorithms, including SimCLR (Chen et al., 2020), MoCo (He et al., 2020) and DINO (Caron et al., 2021), which allows users to potentially utilize large-scale data obtained in real-world scenarios (Wang et al., 2023a).
- **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.
- **Command Line Interface (CLI) Support**: Wespeaker can be easily installed with `python pip`, with access to various pretrained

[1] https://mm.kaist.ac.kr/datasets/voxceleb/voxsrc
[2] http://www.cnceleb.org/competition

models,[3] allowing users to easily obtain off-the-shelf high-quality speaker embeddings. These embeddings can be directly applied to speaker-related tasks such as speech synthesis (Jia et al., 2018) and target speaker extraction (Xu et al., 2020).

Overall, Wespeaker is likely to be of interest to various user groups in different ways:

1. For researchers entering the field, we provide step-by-step tutorials that help them quickly grasp the state-of-the-art systems in this domain.[4]
2. For other researchers in this field, we offer competitive baseline systems, allowing for fair algorithm comparisons. Recipes for various standard datasets such as VoxCeleb and NIST SRE are provided.
3. For industry professionals, Wespeaker supports industrial-scale data management, making it easy for them to train models on their own datasets. Additionally, we provide frameworks for self-supervised learning, which have the potential to leverage large amounts of unlabeled data. Wespeaker also provides deployment support on different platforms, which helps to build fast prototypes.
4. For researchers working on other tasks that require speaker modeling, Wespeaker offers off-the-shelf usage. This includes APIs for extracting embeddings, computing speaker similarity, performing diarization, .etc. We also provide a range of models to choose from, facilitating their direct application to their specific tasks.

In summary, Wespeaker strives to be a comprehensive and user-friendly toolkit for speaker representation learning. By providing an intuitive interface and high-performance models, Wespeaker caters to the needs of diverse users. Our goal is to contribute to advancements in research and applications of speaker embedding learning through this toolkit and its related resources. In this article, we give a thorough introduction to the design principles and functionalities of Wespeaker.

2. Background on deep speaker embedding learning

2.1. Discriminative speaker embedding learning

For a standard deep speaker embedding learning system, the input is frame-level acoustic features (*e.g.*, filter banks) 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 speaker labels. Moreover, a classification loss is adopted to provide a speaker-discriminative supervision signal. A typical architecture is illustrated in Fig. 1, which represents the common structure shared by many mainstream speaker models, including x-vector (Snyder et al., 2018), r-vector (Zeinali et al., 2019; Wang et al., 2019), and ECAPA-TDNN (Desplanques et al., 2020).

We refer to these classification-based methods as Discriminative Speaker Embedding Learning (Wang et al., 2019). Within this framework, speaker labels are crucial for computing the classification loss. This requirement presents challenges in terms of obtaining training datasets and utilizing large-scale in-the-wild data effectively.

2.2. Self-supervised speaker embedding learning

In the development of deep speaker representation learning, another type of method previously referred to as metric learning (Chung et al., 2020a) has emerged. This approach learns representations with speaker identity modeling capability by contrasting positive and negative examples (Zhang et al., 2018; Wan et al., 2018; Huang et al., 2018a). However, the learning methods during that period still required the use of speaker labels for constructing positive and negative pairs. The self-supervised learning methods to be introduced below are quite similar to these methods, but further define positive and negative examples based on a label-free hypothesis: *different segments extracted from the same utterance belong to the same speaker (positive samples), while those from different utterances belong to different speakers (negative samples).*

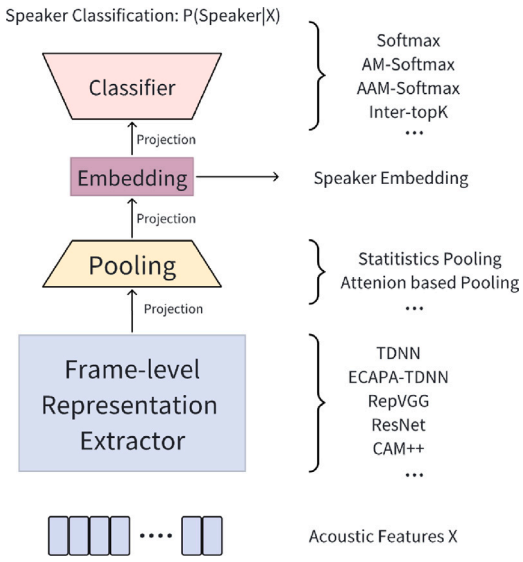

Fig. 1. A typical discriminative deep speaker embedding learning system.

In Wespeaker, we implemented three widely recognized self-supervised learning (SSL) methods including SimCLR (Chen et al., 2020), MoCo (He et al., 2020), and DINO (Caron et al., 2021). The learning frameworks of these three paradigms are illustrated in Fig. 2, and we will provide a brief introduction to each of them in the following sections.

2.2.1. SimCLR

SimCLR (Simple Contrastive Learning of Representations) (Chen et al., 2020) is a self-supervised learning algorithm that uses contrastive learning to learn representations from unlabeled data, originally designed for visual representation learning. In the context of speech representation learning, the specific approach is as follows: To construct a batch consisting of N speech samples, for each speech sample, two random segments are cropped and subjected to different data augmentation techniques. These segments are then fed into the same speaker encoder, producing speaker embeddings. Subsequently, an MLP-based non-linear transformation is applied to remove irrelevant information, yielding representations for optimization (Chen et al., 2020). Based on the assumption that samples originating from the same speech segment are positive pairs, while the remaining $2N - 1$ pairs are negative, the model is optimized using the contrastive loss named InfoNCE (Oord et al., 2018).

2.2.2. MoCo

MoCo (He et al., 2020), short for Momentum Contrast, is another self-supervised learning method that is also built based on contrastive loss. The main idea of MoCo is to build a large and consistent memory bank of data samples and their encoded representations and to minimize the distance between matching pairs of queries and keys while maximizing the distance between non-matching pairs. The memory bank is maintained as a queue of data samples, where the newest samples are enqueued and the oldest ones are dequeued.

[3] https://github.com/wenet-e2e/wespeaker/blob/master/docs/pretrained.md

[4] https://wenet.org.cn/wespeaker/index.html

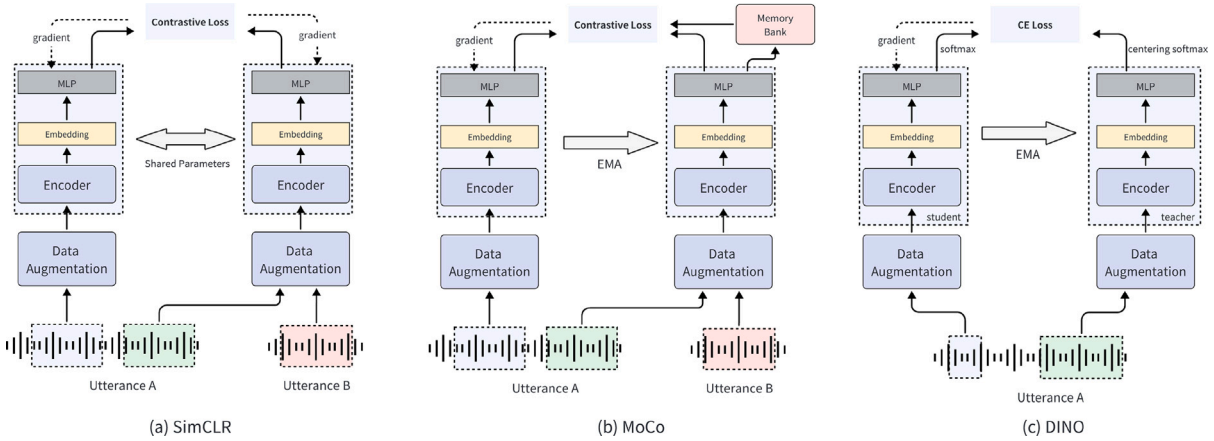

Fig. 2. Frameworks of the Contrastive Self-supervised Learning Algorithms: (a) SimCLR, (b) MoCo, and (c) DINO. Chunks from utterance A form a positive pair, while the chunks from utterance A and utterance B form a negative pair. Note that although the MLP layer is a commonly used setup in the literature, we have made it an optional configuration in Wespeaker since it does not have a significant impact on performance.

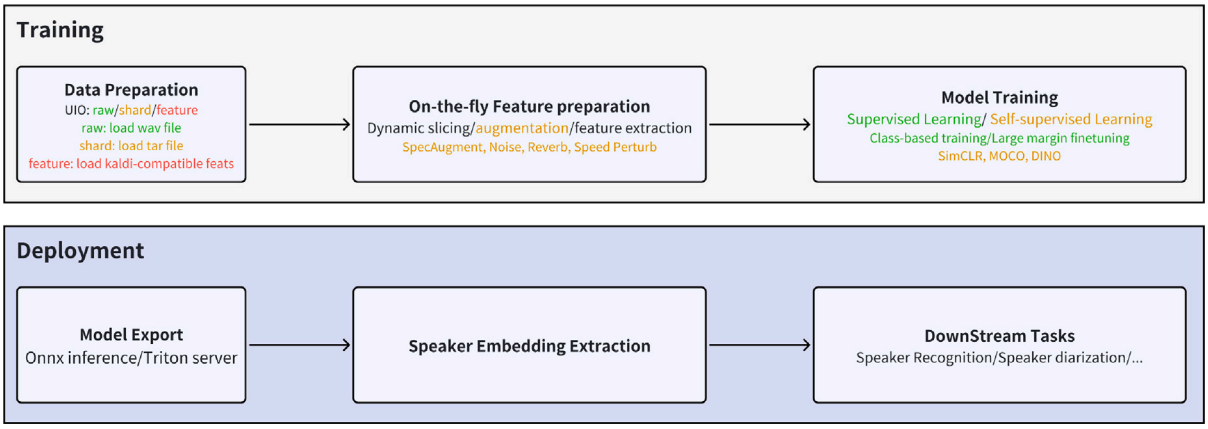

Fig. 3. The overall structure of Wespeaker.

2.2.3. DINO

Different from SimCLR and MoCo, DINO (DIstillation of knowledge with NO labels) (Caron et al., 2021) adopts a different strategy for constructing training samples. Its optimization no longer relies on negative samples but maximizes the similarity between positive sample pairs, thereby avoiding potential errors introduced during the construction of negative samples (such as mistakenly treating samples from the same speaker within a batch as different speakers based on assumptions). As shown in Fig. 2(c), DINO adopts a teacher–student architecture with momentum updates, which is similar to MoCo (He et al., 2020). In addition, positive sample pairs in DINO have unequal-length segments. Longer segments are inputted into the teacher model, and the results are sharpened to serve as the target for model optimization. Shorter segments are inputted into the student model, and gradient backpropagation is performed using cross-entropy loss. In terms of performance, DINO demonstrates significant improvements (Zhang and Yu, 2022; Chen et al., 2023a; Han et al., 2024) compared to the other two frameworks.

3. Wespeaker

3.1. Overall structure

Fig. 3 shows the overall structure and pipeline of Wespeaker. A standard procedure contains data management on the disk, online feature preparation, and model training. Once the model has converged, 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.

3.2. Data management

Wespeaker offers support for three distinct modes of data management: *raw mode*, *shard mode*, and *feat mode*. The feat mode is specifically designed for loading pre-extracted features from the disk, whereas the other two modes, raw and shard, are intended for loading files in wave format. As shown in Fig. 4, these three modes are unified under the Unified IO (UIO) framework within Wespeaker, enabling various data organization and retrieval methods for both small and large datasets. This unified approach ensures efficient data reading, even for industrial-scale datasets.

3.2.1. Raw mode

Firstly, we support loading the raw wave files from the disk directly, which is called “raw mode” in Wespeaker. It is suitable and convenient for small-dataset training or testing cases. The raw mode requires frequent opening and closing of small wave files, which is very IO-consuming and makes training slow, especially for large-scale training sets.

3.2.2. Shard mode

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

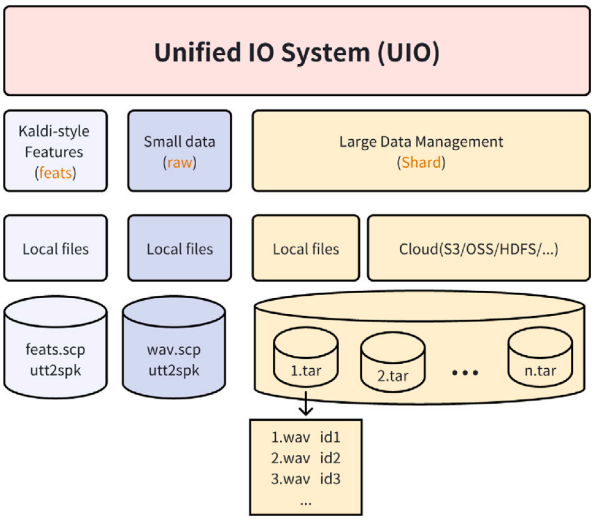

Fig. 4. Unified IO in Wespeaker.

introduce the unified IO (UIO) mechanism of Wenet[5] to the data management in Wespeaker. This mechanism was inspired by the TFRecord format used in Tensorflow (Abadi et al., 2016) and AIStore (Aizman et al., 2019), which packs each set of small files into a bigger shard via the GNU tar. As shown in Fig. 4, 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.

3.2.3. Feat mode

While Wespeaker is primarily designed to be independent of Kaldi, we recognize that many users may have pre-extracted features in Kaldi format with various configurations. In order to facilitate the reuse of these existing features, we provide a "feat mode" to support the loading of Kaldi-style features, thus addressing this common requirement.

3.3. 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, 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. 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.

The supported online processors are summarized in Table 2, while Fig. 5 presents the pipeline of online feature preparation in Wespeaker. Wespeaker notably facilitates effortless configuration for organizing diverse processors into a pipeline, ensuring both efficiency and ease of extension.

3.4. SOTA model implementation

Wespeaker supports different popular speaker embedding learning models, margin-based softmax training objectives, and several pooling functions.

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

Table 2
Description of Online Feature Processors.

Operation	Description
Filter	Filter out very short utterances and randomly chunk over-length utterances.
Local shuffle	Create and shuffle a local buffer for sample randomness across different epochs.
Spk2id	Convert speaker names into index based speaker IDs.
Resample	Resample data to the desired sample rate.
Speed perturb	Adjust the speed of the training data with a certain probability.
Random chunk	Randomly divide the training data into equal-length chunks and apply repeated padding to shorter segments.
VAD	Filter out the non-speech part.
Noise, Reverb	Augment the data by introducing noise or simulating reverberation.
Compute Fbank	Extract filter banks (Fbank) features from raw PCM data.
CMVN	Apply cepstral mean and variance normalization per chunk.
SpecAug	Utilize Spec-Augmentation (Park et al., 2019) on the feature.
Batch	Organize the data into fixed-size batches.

- **TDNN** based x-vector (Snyder et al., 2018), this is a milestone work that leads the following deep speaker embedding era.
- **ResNet** based r-vector and its deeper version, this is the best system of VoxSRC 2019 (Zeinali et al., 2019) and CNSRC 2022 (Chen et al., 2022a).
- **ECAPA-TDNN**, an improved version of TDNN by incorporating channel attention and multi-level feature aggregation, this is the champion system of VoxSRC 2020 (Desplanques et al., 2020).
- **RepVGG** decouples the training time and inference time architecture, resulting in good performance and inference speed. This is the best system of VoxSRC 2021 (Zhao et al., 2021).
- **CAM++**, a modified densely connected time delay neural network (D-TDNN) that utilizes a context-aware masking mechanism (Wang et al., 2023c). It also incorporates a novel multi-granularity pooling technique to capture contextual information at various levels.
- **ERes2Net**, an enhanced Res2Net architecture to improve the robustness of speaker embeddings by aggregating global signals from multi-scale features (Chen et al., 2023b).

Pooling functions aggregate frame-level features into segment-level representations. To be more formulated, N frames of d-dimensional deep features $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n, \dots, \mathbf{x}_N\}$ can be obtained from frame-level transformation, a pooling layer $\mathcal{P}$ is adopted to aggregate $\mathbf{X}$ to a single representation $\mathbf{z}$. Wespeaker supports the statistics-based and attention-based ones.

Statistics-based pooling functions have long been used in speaker embedding learning and can be categorized into three types, based on the statistics they utilize. These include Temporal Average Pooling, denoted as $\mathcal{P}_{\text{TAP}}$, Temporal Standard Deviation Pooling (Wang et al., 2021), represented as $\mathcal{P}_{\text{TSDP}}$, and Temporal Statistics Pooling (Snyder et al., 2018), referred to as $\mathcal{P}_{\text{TSTP}}$. The latter encompasses both first and second-order statistics.

$$\mathcal{P}_{\text{TAP}}(\mathbf{X}) = \boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{1}$$

$$\mathcal{P}_{\text{TSDP}}(\mathbf{X}) = \boldsymbol{\sigma} = \sqrt{\frac{1}{N}\sum_{n}^{N}\mathbf{x}_n \odot \mathbf{x}_n - \boldsymbol{\mu} \odot \boldsymbol{\mu}} \tag{2}$$

$$\mathcal{P}_{\text{TSTP}}(\mathbf{X}) = [\boldsymbol{\mu}^{\top}, \boldsymbol{\sigma}^{\top}]^{\top} \tag{3}$$

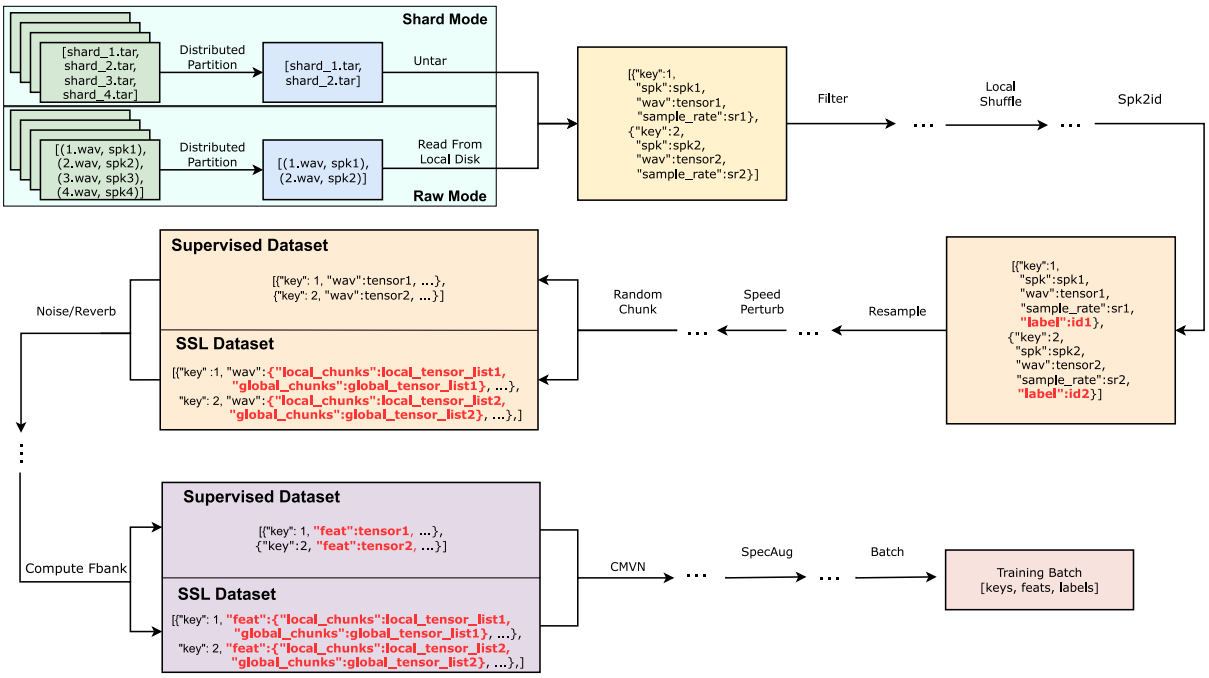

Fig. 5. The pipeline of online feature preparation for both the supervised and self-supervised learning paradigms.

For attention-based pooling methods, we provide the implementation of attentive statistics pooling (ASTP) and its variants: Channel- and context-dependent ASTP (CC-ASTP) (Desplanques et al., 2020), Multi-head ASTP (MH-ASTP) (India et al., 2019) and Multi-query multi-head ASTP (MQMH-ASTP) (Zhao et al., 2022).

Compared to the normal TSTP, ASTP compute the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in the following way:

$$e_n = \mathbf{v}^\top f\left(\mathbf{W}\mathbf{x}_n + \mathbf{b}\right) + k \tag{4}$$

$$\alpha_n = \frac{\exp\left(e_n\right)}{\sum_\tau^N \exp\left(e_\tau\right)} \tag{5}$$

$$\boldsymbol{\mu} = \sum_n^N \alpha_n \mathbf{x}_n \tag{6}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{N}\sum_n^N \alpha_n \mathbf{x}_n \odot \mathbf{x}_n - \boldsymbol{\mu} \odot \boldsymbol{\mu}} \tag{7}$$

where $\mathbf{W}$ and $\mathbf{b}$ represent learnable parameters that are used to transform $\mathbf{x}_n$ into a lower dimension for the subsequent attention computation. The self-attention score, obtained from the non-linear function $f(\cdot)$ (by default, tanh in Wespeaker), is further processed through a linear layer with weights $\mathbf{v}$ and bias k. This transformation results in the scalar score e_t. The other variants follow a similar process but differ in detailed computation. For instance, CC-ASTP involves a channel-dependent parameter computation process. Specifically, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are calculated for each channel, resulting in $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c$. Similarly, intermediate variables such as e_n, k, α_n are computed individually for each channel, yielding e_{nc}, k_c, α_{nc}.

3.5. Training strategies

3.5.1. Training objectives

Loss functions play a crucial role in deep speaker embedding learning. We provide support for various types of loss functions, including the standard softmax cross-entropy loss, as well as different margin-based variants (Hajibabaei and Dai, 2018; Xiang et al., 2019; Han et al., 2023). These variants include A-softmax (Liu et al., 2017; Huang et al., 2018b; Wen et al., 2021), AM-softmax (Wang et al., 2018a), and AAM-softmax (Deng et al., 2019).

In addition to supporting different loss functions, we also provide support for commonly used techniques such as the inter-topk and sub-center algorithms (Zhao et al., 2022). These techniques aim to enhance the discriminative ability of the learned embeddings by considering specific subsets of samples within a mini-batch or using sub-centers to improve intra-class compactness.

3.5.2. Learning rate scheduling

Wespeaker implements the learning rate schedule as the composition of two functions. The variation function of the learning rate with respect to time $lr(t)$ 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:

$$g(t) = \begin{cases} \frac{t}{T_{\text{warm}}}, & t < T_{\text{warm}}. \\ 1, & T_{\text{warm}} \le t < T. \end{cases} \tag{8}$$

$$h(t) = \eta_0 \cdot \exp\left(\frac{t}{T}\ln\left(\frac{\eta_T}{\eta_0}\right)\right) \tag{9}$$

where t, T_{warm}, T represents the current, the warm-up, and the total iterations, η_0 and η_T denotes the initial and final learning rate, respectively.

3.5.3. Margin scheduling

The margin scheduler is a three-stage function $m(t)$:

$$m(t) = \begin{cases} 0, & t < T_1. \\ f(t), & T_1 \le t < T_2. \\ M, & T_2 \le t < T. \end{cases} \tag{10}$$

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

3.5.4. Large margin fine-tuning

The large margin fine-tuning strategy was first proposed in Thienpondt et al. (2021) and widely used in speaker verification challenge systems (Chen et al., 2022a; Zhao et al., 2021; Makarov et al., 2022) 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 6 s training segments.

3.6. Back-end support

For the 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) were more widely used. Wespeaker implements both scoring back-ends.

3.6.1. Two-cov PLDA and unsupervised adaptation

In the two-covariance variant of Probabilistic Linear Discriminant Analysis model, the differences between speakers (Inter-speaker variability) and the differences within a speaker's different audio sessions (Inter-session variability) are described using the across-class covariance matrix $\boldsymbol{\Sigma}_{\text{ac}}$ and the within-class covariance matrix $\boldsymbol{\Sigma}_{\text{wc}}$, respectively. Two-covariance PLDA assumes that the speaker representation $\boldsymbol{\eta}$ is generated through two sampling steps. First, latent variable $\mathbf{z}$ representing the speaker is sampled from the prior distribution:

$$\hat{\mathbf{z}} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}_{\text{ac}}). \tag{11}$$

Then, for each specific speaker represented by $\hat{\mathbf{z}}$, the observations (speaker representations) are samples from within-class normal distribution:

$$\boldsymbol{\eta} \sim p(\boldsymbol{\eta}|\hat{\mathbf{z}}) = \mathcal{N}(\boldsymbol{\eta}; \hat{\mathbf{z}}, \boldsymbol{\Sigma}_{\text{wc}}). \tag{12}$$

For recipes such as NIST SRE, where indomain unlabeled data is provided, we also provide the unsupervised PLDA adaptation implementation. This method essentially estimates the excess variance in the adaptation data and distributes a portion of it to the PLDA across-class covariance matrix and another portion of it to the PLDA within-class covariance matrix.[6] With this feature, we can use unlabeled target domain data for fast domain adaptation, thereby enhancing the model's performance on the test set.

3.6.2. TPSDA

Toroidal Probabilistic Spherical Discriminant Analysis (TPSDA) is a model assuming that the observations live on the unit hypersphere, *i.e.*, the speaker representations are length-normalized which is often the case even when other scoring backends are used. TPSDA can be seen as analogous to PLDA, with Normal distributions replaced by Von Mises–Fisher (VMF). TPSDA has been demonstrated to surpass the traditional PLDA model in performance when applied to discriminatively trained speaker embeddings (Silnova et al., 2023; Brümmer et al., 2022). We perceive this model as having significant potential as a scoring back-end, and consequently, we have integrated its implementation into Wespeaker. However, it is important to note that this is currently more of an experimental feature, which we plan to refine and improve in future iterations.

3.6.3. Score normalization

Adaptive Symmetric Score Normalization (AS-Norm) is implemented in Wespeaker, averaging the normalized scores from Z-Norm and T-Norm (Matejka et al., 2017), to normalize the speaker verification scores.

4. Deployment and product-oriented setups

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, supporting speaker embedding extraction and diarization tasks. Detailed information including the instructions and performance can be found at https://github.com/wenet-e2e/wespeaker/tree/master/runtime/server. Currently, a C++ API library and runnable demos are provided while the users can also implement their customized system by using the C++ library. We support three mainstream platforms, namely Windows, Linux, and mac. Detailed information can be found at https://github.com/wenet-e2e/wespeaker/tree/master/runtime/onnxruntime. For on-device runtime, we support export "onnx" format to "horizon" format, which can be deployed on the horizon X3'PI. Detailed information including the instructions and performance can be found at https://github.com/wenet-e2e/wespeaker/tree/master/runtime/horizonbpu. 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 enables effortless utilization of the pre-trained models for various downstream tasks.[7]

5. 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.

5.1. 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 (Snyder et al., 2015) are used as additive noises, while the simulated room impulse responses (RIRs)[8] 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 10 ms frameshift and a 25 ms frame window. All training data are chunked into 200 frames and CMN (without CVN) is also applied. Note that SpecAug is not used in any experiment. For VoxCeleb and CNCeleb, AAM loss is used as the optimization objective, with the margin scheduler as described in Section 3.5.3, the large-margin finetuning described in Section 3.5.4 is applied with a margin of 0.5 and segmental length of 6 s.

5.2. Speaker verification

For our speaker verification experiments, we have included two publicly available datasets: VoxCeleb and CNCeleb. Additionally, we incorporated NIST SRE, given the longstanding adoption of the SRE series in the speaker recognition community over the past two decades.

It is worth noting that, in our experiments related to VoxCeleb, we aim to present results comprehensively. This includes results from various supervised training models, different self-supervised training methods, as well as comparisons of model parameters and inference speeds across different models. As for CNCeleb and NIST SRE, we provide corresponding basic recipes for reference. Users can easily adapt the more complex models and training patterns used with VoxCeleb to other datasets.

[6] See https://github.com/kaldi-asr/kaldi/blob/master/src/ivector/plda.cc.

[7] 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.

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

Table 3
Supervised results achieved using different architectures on the VoxCeleb dataset, "dev" of part 2 is used as the training set.

Literature/Toolkits	Architecture	Voxceleb1_O		Voxceleb1_E		Voxceleb1_H	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
Desplanques et al. (2020)	ECAPA-TDNN	0.870	0.107	1.120	0.132	2.120	0.210
Zeinali et al. (2019)	ResNet34	1.310	0.154	1.380	0.163	2.500	0.233
AsvSubtools[b]	ECAPA-TDNN	0.856	–	–	–	–	–
	Conformer	0.792	–	–	–	–	–
	ResNet34	1.538	–	1.705	–	2.985	–
	Extended-TDNN[a]	1.729	–	–	–	–	–
SpeechBrain[c]	TDNN[a]	3.23	–	–	–	–	–
	ECAPA-TDNN[a]	0.80	–	–	–	–	–
	ResNet-TDNN[a]	0.95	–	–	–	–	–
	ECAPA-TDNN	1.30	–	1.98	–	3.62	–
Nemo[d]	TDNN[a]	1.96	–	–	–	–	–
	ECAPA-TDNN[a]	0.92	–	–	–	–	–
	titanet_large[a]	0.66	–	–	–	–	–
3d-speaker[e]	CAM++	0.73	0.091	–	–	–	–
	ERes2Net	0.97	0.090	–	–	–	–
Wespeaker	TDNN	1.590	0.166	1.641	0.170	2.726	0.248
	ECAPA-TDNN	0.728	0.099	0.929	0.100	1.721	0.169
	CAM++	0.654	0.087	0.805	0.092	1.576	0.164
	ERes2Net	0.744	0.074	0.896	0.092	1.603	0.151
	RepVGG	0.750	0.083	0.846	0.090	1.495	0.141
	ResNet34	0.723	0.069	0.867	0.097	1.532	0.146
	ResNet50	0.803	0.061	0.887	0.092	1.519	0.136
	ResNet101	0.542	0.052	0.758	0.079	1.398	0.128
	ResNet152	0.495	0.033	0.685	0.069	1.205	0.105
	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**

[a] Models with ∗ are trained with the combined dataset of Vox1 dev and Vox2 dev, while others are trained with only Vox2 dev.
[b] Results from https://github.com/Snowdar/asv-subtools
[c] Results from https://github.com/speechbrain/speechbrain/tree/develop/recipes/VoxCeleb/SpeakerRec
[d] Results from https://github.com/NVIDIA/NeMo/tree/main/examples/speaker_tasks/recognition
[e] Results from https://github.com/alibaba-damo-academy/3D-Speaker/tree/3dspeaker/egs/voxceleb/

For all recipes on the speaker verification task, we demonstrate the results in terms of Equal Error Rate (EER) and minimum detection cost (minDCF) with $P_{\text{tar}} = 0.01$. Scores are post-processed using AS-Norm to boost the systems' performance.

5.2.1. VoxCeleb

VoxCeleb dataset (Nagrani et al., 2020b) was introduced by Oxford University and has emerged as one of the most widely utilized 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.

For all the systems on the VoxCeleb datasets demonstrated in Table 3, cosine similarity is adopted as the scoring backend. Large-margin finetuning and AS-Norm are applied for all Wespeaker systems. The results show that our implementation achieves very competitive numbers compared with the original ones in the literature and other open-source toolkits. Scaling the ResNet deeper can further boost the performance significantly. The best system, ResNet293, achieves impressive performance with EER values of 0.447%, 0.657%, and 1.183% on the three respective evaluation sets. However, it is important to note that despite its strong performance, using such a large model results in increased memory usage, higher computational resource consumption, and longer latency. To provide a more intuitive understanding, we have presented the corresponding parameter count, memory usage, MACs (Multiply-Accumulate operations), and RTF (Real-Time Factor) in Table 4. For all model architectures, only one embedding layer is used and the embedding size is set to 256 for a fair comparison.

Self-supervised learning (SSL) has attracted many researchers in speaker verification because it effectively alleviates the problem of lack of speaker labels. In Wespeaker, we provide open-source implementations of several mainstream SSL frameworks for speaker verification on the VoxCeleb dataset, including SimCLR, MoCo, and the very popular DINO. The results of different frameworks and backbones are presented in Table 5. The negative-pair-free DINO achieves significant performance superiority when compared to contrastive learning-based methods such as SimCLR and MoCo. Furthermore, ECAPA-TDNN(C1024)-based DINO achieved EERs of 2.627%, 2.665%, and 4.644% on three test trials of Voxceleb without using any human-labeled data, which is a very strong and competitive result.

Table 4
Comparison of model parameters, memory cost, MACs, and RTF, the batch-size is set to 16 and frame-number is set to 100.

Architecture	Params (M)	Memory (G)	MACs (G)	RTF
TDNN	3.645	0.055	3.990	0.0127
ECAPA-TDNN(C512)	6.387	0.193	8.320	0.0183
ECAPA-TDNN(C1024)	14.85	0.354	21.23	0.0417
CAM++	7.176	0.701	9.010	0.0229
ERes2Net	7.266	2.427	27.17	0.0525
RepVGG	6.264	0.423	37.38	0.0551
ResNet34	6.634	1.053	36.50	0.0607
ResNet50	11.13	3.484	40.91	0.0732
ResNet101	15.89	5.174	78.79	0.1246
ResNet152	19.81	7.254	116.6	0.1793
ResNet221	23.79	11.20	167.9	0.2675
ResNet293	28.62	15.15	221.4	0.3640

5.2.2. 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 (Li et al., 2022) is similar to the one of VoxCeleb, many recordings are shorter than 2 s in this dataset. Therefore, in the data preparation, we first concatenate the short audios

Table 5
Performance (EER%) of SSL-based systems on the VoxCeleb evaluation set.

Toolkits	Paradigm	Architecture	VoxCeleb1_O	VoxCeleb1_E	VoxCeleb1_H
3d-speaker	RDINO	ECAPA-TDNN (C1024)	3.16	–	–
Wespeaker	SimCLR	ECAPA-TDNN (C512)	8.523	9.417	14.907
		ECAPA-TDNN (C1024)	8.318	8.770	14.665
	MoCo	ECAPA-TDNN (C512)	8.709	9.287	14.756
		ECAPA-TDNN (C1024)	8.473	8.946	14.635
	DINO	ResNet34	3.170	3.324	5.821
	DINO	ECAPA-TDNN (C512)	3.016	3.093	5.538
	DINO	ECAPA-TDNN (C1024)	**2.627**	**2.665**	**4.644**

Table 6
Results on the CNCeleb evaluation set.

Toolkits	Architecture	EER(%)	minDCF
ASVSubtools	ResNet34	9.141	0.463
3D-Speaker	ECAPA-TDNN (C1024)	8.01	0.445
	CAM++	6.78	0.393
	ERes2Net	6.69	0.388
Wespeaker	TDNN	8.960	0.446
	ECAPA-TDNN (C1024)	7.395	0.372
	CAM++	7.052	0.368
	ERes2Net	6.474	0.343
	ResNet34	6.492	0.354
	ResNet221	**5.655**	**0.330**

from the same genre and same speaker to construct audios longer than 5 s.

The results obtained using different backbones are exhibited in Table 6. 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. This method has shown considerable performance improvements in our experiments compared to simply concatenating all enrollment utterances to extract a single embedding.

5.2.3. NIST speaker recognition evaluation (SRE)

The NIST SRE, which began in 1996, is a long-standing speaker recognition evaluation event. Since then, NIST has organized an SRE every 1–2 years to promote research and development in the field. Each evaluation introduces new datasets to adapt to technological advancements and changing application requirements. Over the years, the NIST SRE has accumulated a large collection of telephone channel datasets.

Unlike VoxCeleb scenario, NIST SREs mainly focused on 8k telephone channel data, covering various noise scenarios and languages. There is a significant difference between the training and testing data, making the task more challenging. Compared to the VoxCeleb Recipe, we have made the following modifications:

- **Voice activity detection**: Due to the presence of a large number of silent segments in the SRE data, we added silence detection during the data preprocessing stage to more effectively process the speech data.
- **Pipe reading support**: SRE data is in .sph format, which most speech reading tools do not support. To allow Wespeaker to better read .sph format data, we supported the functionality of reading speech data from pipes during the data preprocessing stage.
- **Unsupervised PLDA adaptation**: In the SRE Recipe, there is a significant domain mismatch between the test data (SRE16/SRE18) and the training data. To improve the performance on the test set, we support the kaldi-style PLDA unsupervised domain adaptation as described in Section 3.6.1.

For the SRE recipe, we followed a similar setup as the ABC systems developed for the NIST SRE21 evaluation (Alam et al., 2022). As training data, we used the CTS Superset (Sadjadi, 2021) plus VoxCeleb 1 dev+test and VoxCeleb 2 dev, downsampled to 8 kHz and passed through the GSM codec provided by SoX.[9] We used the same augmentation settings as for the VoxCeleb recipe. The feature configuration was also the same except that we used 64-dimensional filterbanks instead of 80-dimensional. As preprocessing before the backend, the embeddings were centered, length-normalized, subjected to LDA including a second centering and, finally, length-normalized again. The backends and the LDA transform were trained on the augmented CTS superset, *i.e.*, VoxCeleb was not included. We evaluate the systems on NIST SRE16 (Seyed et al., 2017), SRE18 (Sadjadi et al., 2019), and SRE21 (Sadjadi et al., 2021).

Table 7 presents the results obtained with cosine, PLDA, and TPSDA backends trained on top of the embeddings with dimensionality reduced to 100 by LDA. For SRE16 and SRE18, unlabeled adaptation datasets are available so for them we also present the results obtained with adapted PLDA. The results are similar to results obtained during ABC's NIST SRE21 efforts with similar architectures trained using other toolkits.

Contrary to the findings in Silnova et al. (2023), the results for TPSDA are somewhat worse than for PLDA. This could be because of a different preprocessing chain before the backend. Further investigation will be carried out.

5.3. Speaker diarization

Speaker diarization (SD) is the task of splitting an audio recording into acoustically homogeneous clusters according to the identity of each speaker. Most diarization systems are based on clustering-based methods that typically consist of two steps: segmentation and clustering. The first step aims at partitioning the audio recording into segments, such that each segment only contains one speaker. The second step aims at dividing segments into clusters based on the identity of each speaker, where the spectral clustering method is adopted in Wespeaker.

5.3.1. VoxConverse

VoxConverse dataset (Chung et al., 2020b), a "in the wild" large-scale speaker diarization dataset, was created by the Visual Geometry Group at Oxford. The dataset was collected from YouTube videos with a semi-automatic pipeline and released for the diarization track in VoxSRC 2020.

The VoxConverse recipe shows how to leverage a pre-trained speaker embedding extractor for speaker diarization. The pre-trained ResNet34 model is used to extract speaker embeddings and spectral clustering is applied to cluster the embeddings.

As demonstrated in Table 8, we have achieved robust results on the VoxConverse dev dataset using two types of Voice Activity Detection (VAD). With oracle VAD derived from manual annotations of the dataset, the pipeline achieves a Diarization Error Rate (DER) of 4.2% on the dev set. When employing Silero-VAD (Silero Team, 2021) as the system VAD, the DER is 6.5%. These results highlight the

[9] https://sourceforge.net/projects/sox/

Table 7
Performance comparison of Wespeaker systems on NIST SRE 2016, 2018, and 2021.

System	Loss	Scoring Method	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Eval		SRE21 Eval	
			EER (%)	minDCF	EER (%)	minDCF	EER (%)	minDCF	EER (%)	minDCF
ResNet34	Softmax	Cosine	4.161	0.330	15.98	0.926	7.839	0.528	11.33	0.670
		PLDA	3.467	0.293	16.03	0.977	6.962	0.491	10.10	0.624
		Adapted PLDA	2.741	0.213	8.962	0.614	6.949	0.512	–	–
		TPSDA	3.954	0.330	11.88	0.729	7.221	0.530	10.69	0.695
	AAM-softmax	Cosine	3.809	0.327	13.79	0.868	7.433	0.519	12.60	0.704
		PLDA	3.513	0.310	13.83	0.920	7.291	0.502	12.18	0.692
		Adapted PLDA	3.467	0.289	12.20	0.798	6.344	0.557	–	–
		TPSDA	4.135	0.380	11.88	0.751	7.121	0.541	12.45	0.760
ResNet101	Softmax	Cosine	3.410	0.283	14.41	0.893	7.119	0.491	9.885	0.623
		PLDA	2.840	0.256	14.33	0.961	6.357	**0.458**	9.061	**0.582**
		Adapted PLDA	**2.150**	**0.186**	**8.382**	**0.583**	6.683	0.494	–	–
		TPSDA	3.134	0.291	11.24	0.688	6.393	0.487	**9.050**	0.630
	AAM-softmax	Cosine	2.736	0.257	11.70	0.754	6.411	0.467	11.24	0.671
		PLDA	2.586	0.233	11.91	0.769	6.265	0.450	10.99	0.660
		Adapted PLDA	2.353	0.211	9.598	0.593	7.857	0.513	–	–
		TPSDA	3.104	0.307	10.488	0.698	**6.155**	0.474	10.69	0.722

Table 8
Results on the VoxConverse dataset.

Set	System	VAD	MISS(%)	FA(%)	SC(%)	DER(%)
dev	Chung et al. (2020b)[a]	–	2.4	2.3	3.0	7.7
	Wespeaker	oracle	2.3	0.0	1.9	4.2
		silero	3.8	0.7	2.0	6.5
		pyannote[c]	**2.7**	**0.2**	**1.8**	**4.8**
test	pyannote[b]	pyannote[c]	2.3	1.8	2.7	6.8
	Wespeaker	silero	4.0	2.4	3.5	9.8
		pyannote[c]	**3.2**	**0.7**	**3.0**	**7.0**

[a] Audio–Visual system
[b] https://huggingface.co/pyannote/speaker-diarization-3.1
[c] Access needs token verification, thus not integrated in Wespeaker

effectiveness of the deep speaker embedding learning capabilities of the Wespeaker toolkit. Furthermore, we demonstrate that integrating VAD from pyannote.audio can lead to further performance enhancements, indicating that optimizing VAD is a promising direction for future work.

5.3.2. GPU clustering

To speed up the clustering and diarization process, Cupy (Okuta et al., 2017) and CuML (Raschka et al., 2020) are utilized to perform GPU-based spectral clustering. All operations of speaker embeddings are using Cupy and we use GPU Kmeans algorithm implementation from CuML. Similar performances of the VoxConverse development set can be obtained from our experiments but with about 3× speedup compared with CPU clustering. With GPU clustering, the whole speaker diarization modules can be run in GPUs thus a GPU-based SD inference solution has been provided using NVIDIA Triton Server in Wespeaker.

6. Conclusion and future work

In this work, we introduced Wespeaker, a research and product-oriented speaker embedding learning toolkit. Wespeaker offers an elegant code framework and dataset-based recipes, along with efficient data management capabilities that enable scalability to large-scale industrial datasets. Researchers and students can swiftly construct mainstream models and achieve highly competitive performance, facilitating rapid exploration and iteration in related research. For industry professionals and individuals seeking to leverage speaker embeddings in downstream tasks such as diarization and target speaker extraction, we provide off-the-shelf deployment codes, user-friendly installation instructions, and various pre-trained models to select from.

Wespeaker aims to foster advancements in the field of speaker representation learning and actively meet the demands of speaker modeling in various downstream applications. Since the initial release (Wang et al., 2023b), we are pleased to report that this toolkit has been integrated or utilized in well-known toolkits such as Espnet (Watanabe et al., 2018), and pyannote.audio (Baroudi et al., 2023), it also assists users to achieve impressive rankings in competitions (Park et al., 2023; Du et al., 2023; Baroudi et al., 2023; Yan et al., 2023).

In the future, Wespeaker will prioritize the following directions: effective adaptation of large pre-trained models (Hsu et al., 2021; Chen et al., 2022b; Peng et al., 2023), compression of model size (Liu et al., 2022, 2023), and integration with more speaker-related tasks. Moreover, Wespeaker will try to benefit from other projects within the Wenet Community, exploiting the state-of-the-art modeling strategies from other tasks.[10] Through these focus areas, Wespeaker strives to stay at the forefront of speaker representation learning, address practical deployment challenges, and continue supporting advancements in various speaker-related tasks.

CRediT authorship contribution statement

Shuai Wang: Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Zhengyang Chen:** Writing – review & editing, Validation, Software, Methodology, Investigation, Data curation. **Bing Han:** Writing – review & editing, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Hongji Wang:** Writing – review & editing, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Chengdong Liang:** Writing – original draft, Validation, Software, Investigation. **Binbin Zhang:** Software, Investigation, Conceptualization. **Xu Xiang:** Software, Methodology, Investigation. **Wen Ding:** Software. **Johan Rohdin:** Software, Investigation. **Anna Silnova:** Writing – review & editing, Software, Investigation. **Yanmin Qian:** Writing – review & editing, Supervision. **Haizhou Li:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[10] For instance, recent research show that ASR pre-trained models can greatly boost the performance of SV systems (Liao et al., 2023; Cai et al., 2023).

Data availability

The project is available at https://github.com/wenet-e2e/wespeaker.

Acknowledgments

Shuai Wang is currently supported by Internal Project of Shenzhen Research Institute of Big Data under grant No. J00220230014. The people from SJTU are supported in part by China NSFC projects under Grants 62122050 and 62071288, and in part by Shanghai Municipal Science and Technology Commission Project under Grant 2021SHZDZX0102. We would like to express our gratitude to the Wenet open-source community, whose dedication and collective efforts have played a pivotal role in the success and growth of Wespeaker.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

Aizman, A., Maltby, G., Breuel, T., 2019. High performance I/O for large scale deep learning. In: 2019 IEEE International Conference on Big Data. Big Data, IEEE, pp. 5965–5967.

Alam, J., Beneš, R., Beszédeš, M., Burget, L., Dahmane, M., andHamed Ghodrati, A.F., Glembek, O., Kang, W.H., Pavel, Matějka, L.M., Plchot, O., Rohdin, J., Silnova, A., Stafylakis, T., 2022. Development of ABC systems for the 2021 edition of NIST speaker recognition evaluation. In: The Speaker and Language Recognition Workshop. Odyssey 2022, ISCA, pp. 346–353.

Baroudi, S., Bredin, H., Plaquet, A., Pellegrini, T., 2023. pyannote. audio speaker diarization pipeline at VoxSRC 2023.

Brown, A., Huh, J., Chung, J.S., Nagrani, A., Zisserman, A., 2022. VoxSRC 2021: The third VoxCeleb speaker recognition challenge. arXiv preprint arXiv:2201.04583.

Brümmer, N., Swart, A., Mošner, L., Silnova, A., Plchot, O., Stafylakis, T., Burget, L., 2022. Probabilistic spherical discriminant analysis: An alternative to plda for length-normalized embeddings. arXiv preprint arXiv:2203.14893.

Cai, D., Wang, W., Li, M., 2021. An Iterative Framework for Self-Supervised Deep Speaker Representation Learning. IEEE, pp. 6728–6732.

Cai, D., Wang, W., Li, M., Xia, R., Huang, C., 2023. Pretraining conformer with asr for speaker verification. In: ICASSP 2023. IEEE, pp. 1–5.

Cai, Z., Zhang, C., Li, M., 2020. From speaker verification to multispeaker speech synthesis, deep transfer with feedback constraint. arXiv preprint arXiv:2005.04587.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A., 2021. Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9650–9660.

Chen, T., Kornblith, S., Norouzi, M., Hinton, G., 2020. A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning. PMLR, pp. 1597–1607.

Chen, Z., Liu, B., Han, B., Zhang, L., Qian, Y., 2022a. The SJTU X-LANCE lab system for CNSRC 2022. arXiv preprint arXiv:2206.11699.

Chen, Z., Qian, Y., Han, B., Qian, Y., Zeng, M., 2023a. A comprehensive study on self-supervised distillation for speaker representation learning. In: 2022 IEEE Spoken Language Technology Workshop. SLT, IEEE, pp. 599–604.

Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., et al., 2022b. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. IEEE J. Sel. Top. Sign. Proces. 16 (6), 1505–1518.

Chen, Y., Zheng, S., Wang, H., Cheng, L., Chen, Q., Qi, J., 2023b. An enhanced Res2Net with local and global feature fusion for speaker verification. arXiv preprint arXiv:2305.12838.

Cho, J., Villalba, J., Moro-Velazquez, L., Dehak, N., 2022. Non-contrastive self-supervised learning for utterance-level information extraction from speech. IEEE J. Sel. Top. Sign. Proces. 16 (6), 1284–1295.

Chung, J.S., Huh, J., Mun, S., Lee, M., Heo, H.-S., Choe, S., Ham, C., Jung, S., Lee, B.-J., Han, I., 2020a. In defence of metric learning for speaker recognition. In: Proc. Interspeech 2020. pp. 2977–2981. http://dx.doi.org/10.21437/Interspeech.2020-1064.

Chung, J.S., Huh, J., Nagrani, A., Afouras, T., Zisserman, A., 2020b. Spot the conversation: speaker diarisation in the wild. arXiv preprint arXiv:2007.01216.

Chung, J.S., Nagrani, A., Coto, E., Xie, W., McLaren, M., Reynolds, D.A., Zisserman, A., 2019. VoxSRC 2019: The first VoxCeleb speaker recognition challenge. arXiv preprint arXiv:1912.02522.

Cooper, E., Lai, C.I., Yasuda, Y., Fang, F., Wang, X., Chen, N., Yamagishi, J., 2020. Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 6184–6188.

Deng, J., Guo, J., Niannan, X., Zafeiriou, S., 2019. ArcFace: Additive angular margin loss for deep face recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition. CVPR.

Desplanques, B., Thienpondt, J., Demuynck, K., 2020. ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. In: Proc. Interspeech. pp. 3830–3834.

Du, M., Fang, X., Li, J., 2023. ChinaTelecom system description to VoxCeleb speaker recognition challenge 2023. arXiv preprint arXiv:2308.08181.

Hajibabaei, M., Dai, D., 2018. Unified hypersphere embedding for speaker recognition. arXiv preprint arXiv:1807.08312.

Han, B., Chen, Z., Qian, Y., 2022. Self-supervised speaker verification using dynamic loss-gate and label correction. pp. 4780–4784. http://dx.doi.org/10.21437/Interspeech.2022-742.

Han, B., Chen, Z., Qian, Y., 2023. Exploring binary classification loss for speaker verification. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 1–5.

Han, B., Chen, Z., Qian, Y., 2024. Self-supervised learning with cluster-aware-DINO for high-performance robust speaker verification. IEEE/ACM Trans. Audio Speech Lang. Process. 32, 529–541. http://dx.doi.org/10.1109/TASLP.2023.3331949.

He, K., Fan, H., Wu, Y., Xie, S., Girshick, R., 2020. Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738.

Hsu, W.-N., Bolte, B., Tsai, Y.-H.H., Lakhotia, K., Salakhutdinov, R., Mohamed, A., 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM Trans. Audio Speech Lang. Process. 29, 3451–3460.

Huang, Z., Wang, S., Qian, Y., 2018a. Joint i-vector with end-to-end system for short duration text-independent speaker verification. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 4869–4873.

Huang, Z., Wang, S., Yu, K., 2018b. Angular softmax for short-duration text-independent speaker verification. In: Proc. Interspeech. pp. 3623–3627.

India, M., Safari, P., Hernando, J., 2019. Self multi-head attention for speaker recognition. arXiv preprint arXiv:1906.09890.

Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Lopez Moreno, I., Wu, Y., et al., 2018. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. Adv. Neural Inf. Process. Syst. 31.

Larcher, A., Lee, K.A., Ma, B., Li, H., 2012. The RSR2015: Database for text-dependent speaker verification using multiple pass-phrases. In: Annual Conference of the International Speech Communication Association. Interspeech.

Li, L., Liu, R., Kang, J., Fan, Y., Cui, H., Cai, Y., Vipperla, R., Zheng, T.F., Wang, D., 2022. CN-celeb: multi-genre speaker recognition. Speech Commun. 137, 77–91.

Liao, D., Jiang, T., Wang, F., Li, L., Hong, Q., 2023. Towards a unified conformer structure: from asr to asv task. In: ICASSP 2023. IEEE, pp. 1–5.

Liu, B., Chen, Z., Wang, S., Wang, H., Han, B., Qian, Y., 2022. DF-RESNet: Boosting speaker verification performance with depth-first design.

Liu, T., Lee, K.A., Wang, Q., Li, H., 2023. Golden gemini is all you need: Finding the sweet spots for speaker verification. arXiv preprint arXiv:2312.03620.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L., 2017. Sphereface: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 212–220.

Lu, H., Wu, Z., Dai, D., Li, R., Kang, S., Jia, J., Meng, H., 2019. One-shot voice conversion with global speaker embeddings. In: Interspeech. pp. 669–673.

Makarov, R., Torgashov, N., Alenin, A., Yakovlev, I., Okhotnikov, A., 2022. ID R&D system description to VoxCeleb speaker recognition challenge 2022.

Matejka, P., Novotnỳ, O., Plchot, O., Burget, L., Sánchez, M.D., Cernockỳ, J., 2017. Analysis of score normalization in multilingual speaker recognition. In: Proc. Interspeech. pp. 1567–1571.

Nagrani, A., Chung, J.S., Huh, J., Brown, A., Coto, E., Xie, W., McLaren, M., Reynolds, D.A., Zisserman, A., 2020a. Voxsrc 2020: The second voxceleb speaker recognition challenge. arXiv preprint arXiv:2012.06867.

Nagrani, A., Chung, J.S., Xie, W., Zisserman, A., 2020b. Voxceleb: Large-scale speaker verification in the wild. Comput. Speech Lang. 60, 101027.

Okuta, R., Unno, Y., Nishino, D., Hido, S., Loomis, C., 2017. CuPy: A numpy-compatible library for NVIDIA GPU calculations. In: Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems. NIPS, URL http://learningsys.org/nips17/assets/papers/paper_16.pdf.

Oord, A.v.d., Li, Y., Vinyals, O., 2018. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

Panayotov, V., Chen, G., Povey, D., Khudanpur, S., 2015. Librispeech: an asr corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 5206–5210.

Park, D.S., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E.D., Le, Q.V., 2019. Specaugment: A simple data augmentation method for automatic speech recognition. arXiv preprint arXiv:1904.08779.

Park, D., Kim, J.W., Kim, K.R., Lee, D.H., Kim, H.K., 2023. GIST-AiTeR speaker diarization system for VoxCeleb speaker recognition challenge (VoxSRC) 2023. arXiv preprint arXiv:2308.07788.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. 32.

Peng, J., Stafylakis, T., Gu, R., Plchot, O., Mošner, L., Burget, L., Černockỳ, J., 2023. Parameter-efficient transfer learning of pre-trained transformer models for speaker verification using adapters. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 1–5.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al., 2011. The kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society.

Raschka, S., Patterson, J., Nolet, C., 2020. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. arXiv preprint arXiv:2002.04803.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., et al., 2021. SpeechBrain: A general-purpose speech toolkit. arXiv preprint arXiv:2106.04624.

Sadjadi, O., 2021. NIST SRE CTS Superset: A Large-Scale Dataset for Telephony Speaker Recognition. NIST SRE website, URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=933116.

Sadjadi, O., Greenberg, C., Singer, E., Mason, L., Reynolds, D., 2021. NIST 2021 Speaker Recognition Evaluation Plan. NIST SRE, URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=932697.

Sadjadi, O., Greenberg, C., Singer, E., Reynolds, D., Mason, L., Hernandez-Cordero, J., 2019. The 2018 NIST speaker recognition evaluation. In: INTERSPEECH. Graz, AT, URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=927673.

Seyed, Kheyrkhah, T., Tong, A., Greenberg, C., Olson, D., Singer, E., Mason, L., Hernandez-Cordero, J., 2017. The 2016 NIST speaker recognition evaluation. In: Interspeech 2017. Stockholm, -1, URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=922849.

Silero Team, 2021. Silero VAD: pre-trained enterprise-grade voice activity detector (VAD), number detector and language classifier. GitHub repository, GitHub, https://github.com/snakers4/silero-vad.

Silnova, A., Brümmer, N., Swart, A., Burget, L., 2023. Toroidal probabilistic spherical discriminant analysis. In: ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 1–5. http://dx.doi.org/10.1109/ICASSP49357.2023.10095580.

Snyder, D., Chen, G., Povey, D., 2015. MUSAN: A music, speech, and noise corpus. arXiv:1510.08484.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S., 2018. X-vectors: Robust dnn embeddings for speaker recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 5329–5333.

Thienpondt, J., Desplanques, B., Demuynck, K., 2021. 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, pp. 5814–5818.

Tong, F., Zhao, M., Zhou, J., Lu, H., Li, Z., Li, L., Hong, Q., 2021. ASV-subtools: Open source toolkit for automatic speaker verification. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 6184–6188.

Variani, E., Lei, X., McDermott, E., Moreno, I.L., Gonzalez-Dominguez, J., 2014. Deep neural networks for small footprint text-dependent speaker verification. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 4052–4056.

Wan, L., Wang, Q., Papir, A., Moreno, I.L., 2018. Generalized end-to-end loss for speaker verification. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 4879–4883.

Wang, S., Bai, Q., Liu, Q., Yu, J., Chen, Z., Han, B., Qian, Y., Li, H., 2023a. Leveraging in-the-wild data for effective self-supervised pretraining in speaker recognition. arXiv preprint arXiv:2309.11730.

Wang, F., Cheng, J., Liu, W., Liu, H., 2018a. Additive margin softmax for face verification. IEEE Signal Process. Lett. 25 (7), 926–930.

Wang, S., Huang, Z., Qian, Y., Yu, K., 2019. Discriminative neural embedding learning for short-duration text-independent speaker verification. IEEE/ACM Trans. Audio Speech Lang. Process. 27 (11), 1686–1696.

Wang, H., Liang, C., Wang, S., Chen, Z., Zhang, B., Xiang, X., Deng, Y., Qian, Y., 2023b. Wespeaker: A research and production oriented speaker embedding learning toolkit. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 1–5.

Wang, Q., Muckenhirn, H., Wilson, K., Sridhar, P., Wu, Z., Hershey, J., Saurous, R.A., Weiss, R.J., Jia, Y., Moreno, I.L., 2018b. Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking. arXiv preprint arXiv:1810.04826.

Wang, S., Yang, Y., Qian, Y., Yu, K., 2021. Revisiting the statistics pooling layer in deep speaker embedding learning. In: 2021 12th International Symposium on Chinese Spoken Language Processing. ISCSLP, IEEE, pp. 1–5.

Wang, H., Zheng, S., Chen, Y., Cheng, L., Chen, Q., 2023c. CAM++: A fast and efficient network for speaker verification using context-aware masking. arXiv preprint arXiv:2303.00332.

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N.E.Y., Heymann, J., Wiesner, M., Chen, N., et al., 2018. ESPnet: End-to-end speech processing toolkit. In: Proc. Interspeech. pp. 2207–2211.

Wen, Y., Liu, W., Weller, A., Raj, B., Singh, R., 2021. Sphereface2: Binary classification is all you need for deep face recognition. arXiv preprint arXiv:2108.01513.

Xiang, X., Wang, S., Huang, H., Qian, Y., Yu, K., 2019. 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, pp. 1652–1656.

Xu, C., Rao, W., Chng, E.S., Li, H., 2020. Spex: Multi-scale time domain speaker extraction network. IEEE/ACM Trans. Audio Speech Lang. Process. 28, 1370–1384.

Yan, X., Yang, Y., Guo, Z., Peng, L., Xie, L., 2023. The NPU-elevoc personalized speech enhancement system for icassp2023 DNS challenge. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 1–2.

Yao, Z., Wu, D., Wang, X., Zhang, B., Yu, F., Yang, C., Peng, Z., Chen, X., Xie, L., Lei, X., 2021. WeNet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit. In: Proc. Interspeech.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al., 2002. The HTK Book, Vol. 3, No. 175. Cambridge university engineering department, p. 12.

Zeinali, H., Wang, S., Silnova, A., Matějka, Plchot, O., 2019. But system description to voxceleb speaker recognition challenge 2019. arXiv preprint arXiv:1910.12592.

Zhang, C., Koishida, K., Hansen, J.H., 2018. Text-independent speaker verification based on triplet convolutional neural network embeddings. IEEE/ACM Trans. Audio Speech Lang. Process. 26 (9), 1633–1644.

Zhang, B., Wu, D., Peng, Z., Song, X., Yao, Z., Lv, H., Xie, L., Yang, C., Pan, F., Niu, J., 2022. WeNet 2.0: More productive end-to-end speech recognition toolkit. In: Proc. Interspeech. pp. 1661–1665.

Zhang, C., Yu, D., 2022. C3-DINO: Joint contrastive and non-contrastive self-supervised learning for speaker verification. IEEE J. Sel. Top. Sign. Proces. 16 (6), 1273–1283.

Zhao, M., Ma, Y., Ding, Y., Zheng, Y., Liu, M., Xu, M., 2022. Multi-query multi-head attention pooling and inter-topk penalty for speaker verification. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, IEEE, pp. 6737–6741.

Zhao, M., Ma, Y., Liu, M., Xu, M., 2021. The speakin system for voxceleb speaker recognition challange 2021. arXiv preprint arXiv:2109.01989.

Zhao, X., Wang, S., Chao, Y., Wu, Z., Meng, H., 2023. Adversarial speaker disentanglement using unannotated external data for self-supervised representation based voice conversion. arXiv preprint arXiv:2305.09167.

Zheng, S., Cheng, L., Chen, Y., Wang, H., Chen, Q., 2023. 3D-speaker: A large-scale multi-device, multi-distance, and multi-dialect corpus for speech representation disentanglement. arXiv preprint arXiv:2306.15354.

Zmolikova, K., Delcroix, M., Ochiai, T., Kinoshita, K., Černockỳ, J., Yu, D., 2023. Neural target speech extraction: An overview. IEEE Signal Process. Mag. 40 (3), 8–29.