Speaker Representation Learning

Theories, Applications and Practice

Shuai Wang[1], Bing Han[2]

[1]Shenzhen Research Institute of Big Data, [2]Shanghai Jiao Tong University

December, 2023

深圳市大数据研究院
Shenzhen Research Institute of Big Data

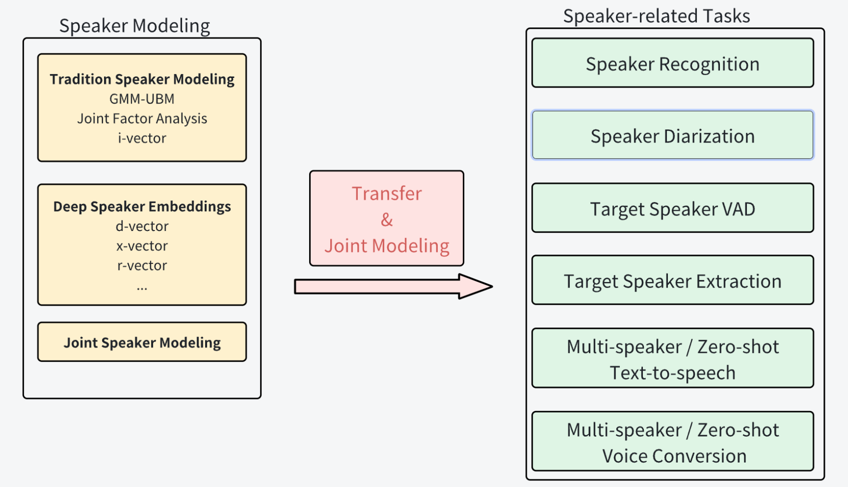
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Outlines

Speaker Modeling

Speaker modeling aims to capture information related to the identity of the speaker while neglecting other attributes.

Speaker modeling in different tasks



Speaker Verification: My voice is my password

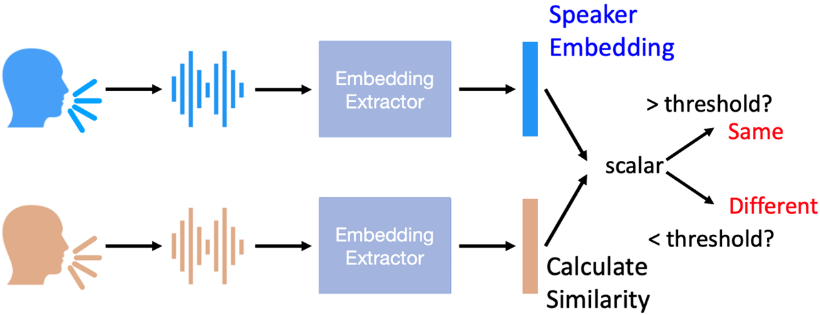
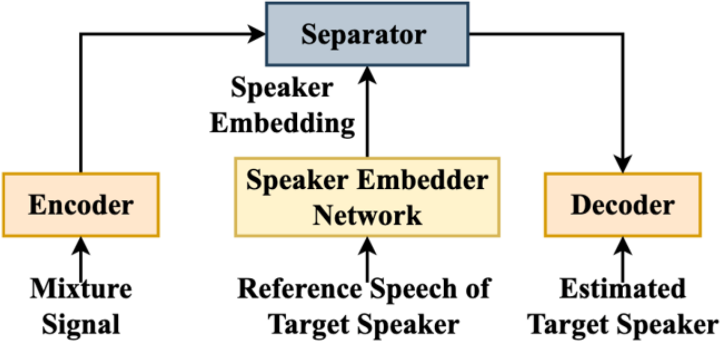


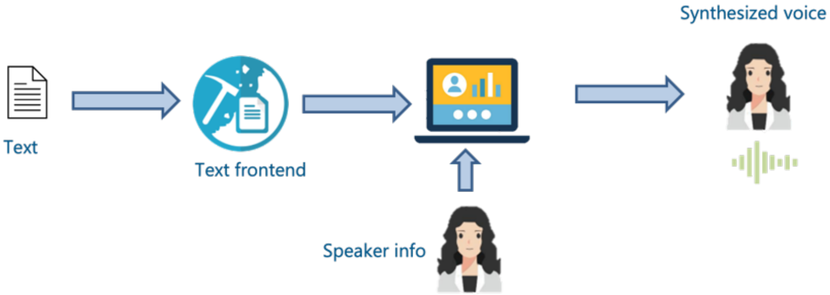
Figure adapted from Hung-yi Lee's DLHLP20 slides[1]

[1] https://speech.ee.ntu.edu.tw/~tlkagk/courses/DLHLP20/

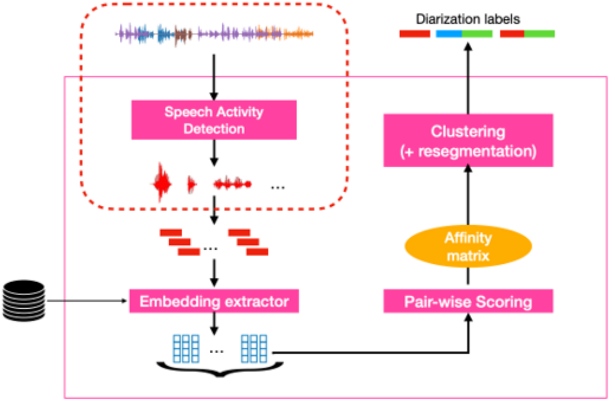
Target Speech Extraction: I only hear your voice



Speaker modeling for the target voice.



Speaker diarization: Who Spoke When?



4

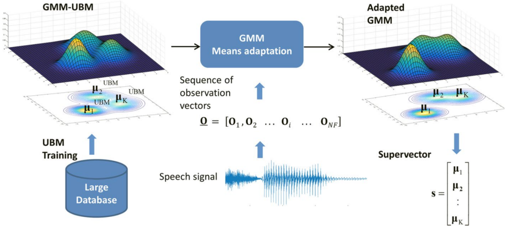


Figure: GMM-UBM[a] model for speaker modeling

[a] Zheng, Zhang, and Xu, "Text-independent speaker identification using gmm-ubm and frame level likelihood normalization".

Gaussian Mixture Model (GMM)[a]

- $p(\mathbf{x}) = \sum_1^K c_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$ s.t. $\sum_1^K c_k = 1$
- Any distribution can be approximated by a weighted linear combination of several Gaussian distributions
- When using GMM to model the speaker's acoustic features, the number of Gaussians can be considered as the types of sounds produced.

Universal Background Model (UBM)

- Usually, a person's registered voice is limited (a few seconds), making it difficult to train a GMM with this data.
- UBM on a large-scale dataset can be trained first and then adapted to a specific speaker's data.

GMM-Supervector

- Concatenate the mean vector of each gaussian to represent the speaker

[a] Reynolds et al., "Gaussian mixture models."

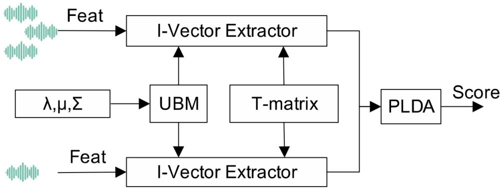


Figure: Block diagram of speaker recognition system based on i-vector

Drawbacks of GMM-Supervector

- Supervectors are extremely high-dimensional (often tens of thousands of dimensions), making them computationally challenging

Decompose Supervector to low-dimensional i-vector[a]:

$$M(s) = m + Tw(s)$$

- $M(s)$: GMM-Supervector for speaker s
- m: speaker-independent supervector
- T: total variability matrix, capturing all sources of variability in the voice samples (both speaker-related and channel-related)
- $w(s)$: i-vector for speaker s

[a] Dehak et al., "Front-end factor analysis for speaker verification".

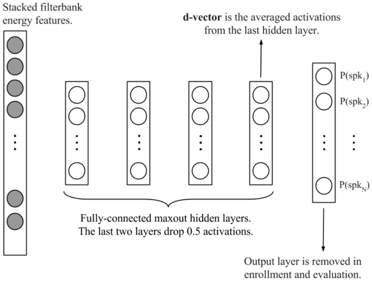


Figure: Architecture of d-vector

d-vector[a] stands out as

- Very early attempts at applying deep neural networks to speaker information modeling
- Demonstrated a good complementarity with i-vector

[a] Variani et al., "Deep neural networks for small footprint text-dependent speaker verification".

Table: Architecture of TDNN based speaker embedding extractor, T denotes the sequence length, N is the number of speakers

Layer	**Layer context**	**Input $\times$ output**
frame1	$[t-2, t+2]$	200 $\times$ 512
frame2	$\{t-2, t, t+2\}$	1536 $\times$ 512
frame3	$\{t-3, t, t+3\}$	1536 $\times$ 512
frame4	$\{t\}$	512 $\times$ 512
frame5	$\{t\}$	512 $\times$ 1500
stats pooling	$[0, T]$	1500 $\times$ 3000
segment1	$\{0\}$	3000 $\times$ 512
segment2	$\{0\}$	512 $\times$ 512
projection	$\{0\}$	512 $\times$ N

x-vector[a] stands out as

- The first deep speaker embedding that beats traditional methods on well-recognized datasets (NIST SRE)
- The first work that introduces segment-level optimization
- Powerful variant ECAPA-TDNN[b]

[a] Snyder et al., "X-vectors: Robust dnn embeddings for speaker recognition".

[b] Desplanques, Thienpondt, and Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification".

Table: Architecture of ResNet34 based speaker embedding extractor, T denotes the sequence length, N is the number of speakers

Layer name	**Structure**	**Output**
Input	–	$40 \times T \times 1$
Conv2D-1	3×3, Stride 1	$40 \times T \times 32$
ResNetBlock-1	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$, Stride 1	$40 \times T \times 32$
ResNetBlock-2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$, Stride 2	$20 \times \frac{T}{2} \times 64$
ResNetBlock-3	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$, Stride 2	$10 \times \frac{T}{4} \times 128$
ResNetBlock-4	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$, Stride 2	$5 \times \frac{T}{8} \times 256$
StatsPooling	–	10×256
Flatten	–	2560
Dense	–	256
Projection	–	N

r-vector[a] stands out as

- The winner system of both tracks of VoxSRC 2019.
- Utilized as the embeddings in the winner systems of all 4 tracks in DIHARD 2019.

[a] Zeinali et al., "But system description to voxceleb speaker recognition challenge 2019".

Labeled data:

- Labeling the data is costly
- Automatically collected data using speaker information, like voxceleb, suffering from the privacy issue
  - The voxceleb dataset is no longer accessible from the official website

Unlabeled data:

- Easily to obtain
- Covers a wider range of real data
- No privacy issue

Training paradigm: Supervised to Unsupervised/Semi-supervised/Self-supervised

Speaker Representation Learning: Trends

Model: From shallow to deep

- GMM, i-vector can be regarded as single-layer MLP
- d-vector, j-vector, x-vector: less than 10 layers.
- ResNet based models (common setup: 34 layers, extended to 293 layers or even 500+ layers in challenges)

- Pure audio modality
- Audio-visual speaker embedding

Speaker Representation Learning: Trends

Paradigm: Training from scratch to leveraging pretrained models

- Training speaker discrinative models from scratch
- Leveraging large pretrained speech models such as WavLM
- Semi-supervised: Finetuning on the DINO models
- Semi-supervised: Iteratively clustering and supervised finetuning

Speaker Representation Learning: Trends

Task: From single task to cross-task

- Pretrained embeddings to be used in different tasks
- Explict joint optimization with the specific task
- Implict speaker modeling in related tasks

Outlines

Optimize the neural network towards the speaker classification direction.

Compared with the classic phoneme classification based ASR system:

ASR: **Close-set problem**, the classes are the pre-defined phonemes/senones in the inference stage

Speaker: **Open-set problem**, usually we assume the speakers in the inference stage are not present in the training set

Discriminative Loss Functions

Classification based loss functions provide discriminative supervision signals, while margin-based ones[2][3] are supervior to the standard softmax-based one[4].

$$L_{\text{softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\mathbf{W}_{y_i}^T\mathbf{x}_i+\mathbf{b}_{y_i}}}{\sum_{j=1}^{c} e^{\mathbf{W}_j^T\mathbf{x}_i+\mathbf{b}_j}} \tag{1}$$

$$L_{\text{AAM-Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{Z} \tag{2}$$

[2] Huang, Wang, and Yu, "Angular Softmax for Short-Duration Text-independent Speaker Verification."

[3] Cai, Chen, and Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system".

[4] Xiang et al., "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition".

The margin-based loss only enlarges the inter-speaker distance.

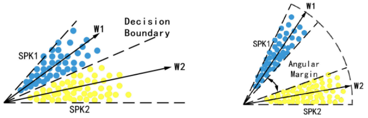


Figure: The decision boundary change after adding the margin-based loss.

An extra center loss[a][b][c] can be applied to minimize the within-class variance.

$$
\begin{aligned}
L &= L_{\text{AAM-Softmax}} + L_C \\
&= L_{\text{AAM-Softmax}} + \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{c}_{y_i}\|^2
\end{aligned}
$$

[a] Cai, Chen, and Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system".

[b] Li et al., "Deep Discriminative Embeddings for Duration Robust Speaker Verification."

[c] Wang et al., "Discriminative neural embedding learning for short-duration text-independent speaker verification".

Outlines

- Leveraging large pre-text pretraining models.
  - Self-supervised Pretrained Speech Models
  - ASR Model Initialization
  - Efficient Finetuning
- Self-supervised Learning Approach
  - SimCLR/MoCo/DINO
  - Stage-wise Iterative Training

- Self-Supervised Pretrained Speech Models
  - Wav2Vec[a]
  - HuBERT[b]
  - WavLM[c]
  - UniSpeech[d]

[a] Baevski et al., "wav2vec 2.0: A framework for self-supervised learning of speech representations".

[b] Hsu et al., "Hubert: Self-supervised speech representation learning by masked prediction of hidden units".

[c] Chen et al., "Wavlm: Large-scale self-supervised pre-training for full stack speech processing".

[d] Chen et al., "Unispeech-sat: Universal speech representation learning with speaker aware pre-training".

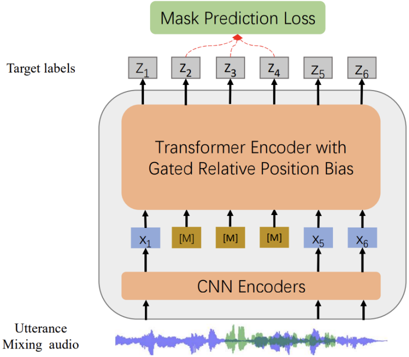


Figure: Model Architecture of WavLM

Finetuning **SSL Speech Models** on Speaker Verification Task[a]

- Replace Fbank with representation from pre-trained models.
- Learnable weighted sum

[a] Chen et al., "Large-scale self-supervised speech representation learning for automatic speaker verification".

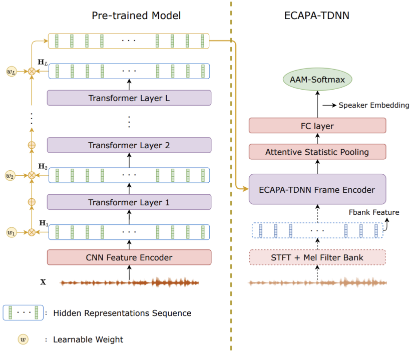


Figure: Leverage Representations from Pre-trained Model

Efficient finetuning Self-supervised Model with adapters on Speaker Verification[a]

- Frozen the large pretrained model
- Use adapters for efficient finetuning on speaker tasks.

[a] Peng et al., "Parameter-efficient transfer learning of pre-trained Transformer models for speaker verification using adapters".

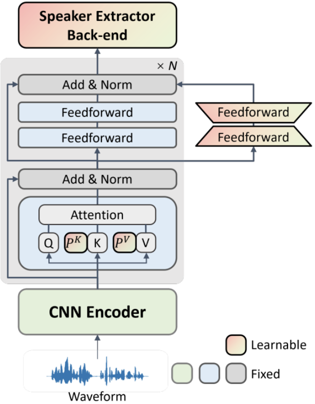


Figure: Schematic diagram of efficient finetuning

Finetuning **ASR Models** on Speaker Verification Task[a][b]

- Pre-train model with ASR dataset.
- Initialize for speaker task training.

[a] Liao et al., "Towards a unified conformer structure: from asr to asv task".

[b] Cai et al., "Pretraining Conformer with ASR for Speaker Verification".

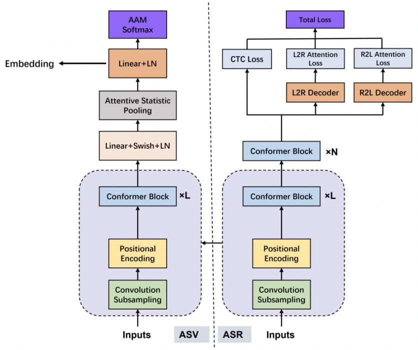


Figure: Schematic diagram of ASR transferring

Assumption of self-supervised learning on Speaker Verification Task.[a]

- Segments from same utterances belong to same speaker.
- Segments from different utterances belong to different speakers.

[a] Huh et al., "Augmentation adversarial training for self-supervised speaker recognition".

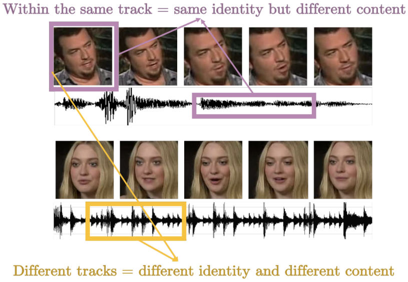


Figure: Schematic diagram of assumption.

Metric learning based loss functions provide contrastive supervision signals, such as Triplet, Prototypical, GE2E[5]and Angular Prototypical[6].

$$L_{\text{Triplet}} = \frac{1}{N}\sum_{j=1}^{N} \max(0, \|\mathbf{x}_{j,0} - \mathbf{x}_{j,1}\|_2^2 - \|\mathbf{x}_{j,0} - \mathbf{x}_{k\neq j,1}\|_2^2 + m) \tag{3}$$

$$L_{\text{Prototypical}} = -\frac{1}{N}\sum_{j=1}^{N} \log \frac{e^{\mathbf{S}_{j,j}}}{\sum_{k=1}^{N} e^{\mathbf{S}_{j,k}}}, where\ \mathbf{S}_{j,k} = \|\mathbf{x}_{j,M} - \mathbf{c}_k\|_2^2 \tag{4}$$

[5] Wan et al., "Generalized end-to-end loss for speaker verification".
[6] Chung et al., "In defence of metric learning for speaker recognition".

Based on SimCLR[a] framework, adapt to speaker task[b]

- Crop two segments from utterance and construct the positive and negative pairs.
- Use metric loss to attract the positive pairs and repel the negative pairs.

[a] Chen et al., "A simple framework for contrastive learning of visual representations".

[b] Zhang, Zou, and Wang, "Contrastive self-supervised learning for text-independent speaker verification".

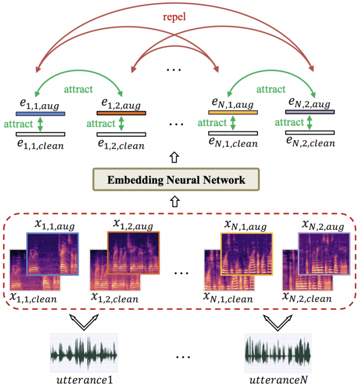


Figure: Schematic diagram of simclr on speaker task.

Based on DINO[a] framework, adapt to speaker task[bc]

- Crop several segments from one utterance and only construct the positive pairs.
- Use cross entropy loss to attract the positive pairs.

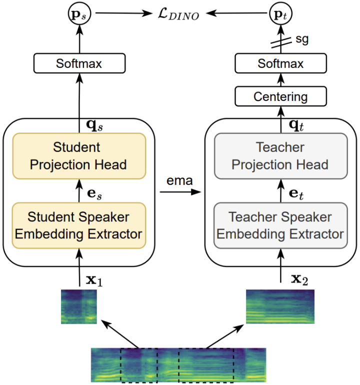


Figure: Schematic diagram of DINO on speaker task.

[a] Caron et al., "Emerging properties in self-supervised vision transformers".

[b] Han, Chen, and Qian, "Self-supervised speaker verification using dynamic loss-gate and label correction".

[c] Chen et al., "A comprehensive study on self-supervised distillation for speaker representation learning".

Two stages based iterative framework[a][b][c].

- I: Contrastive training
- II: Iterative clustering and representation learning.

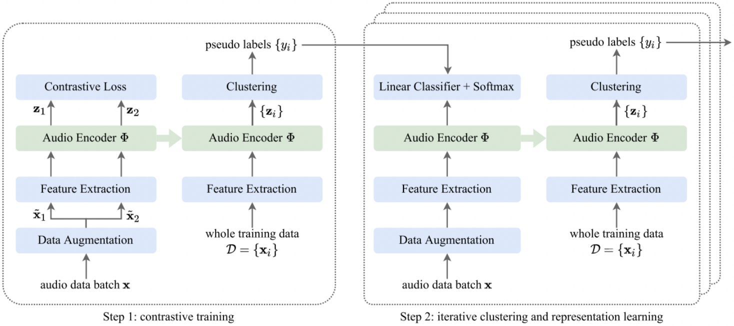


Figure: Schematic diagram of iterative framework for SSL speaker verification.

[a] Cai, Wang, and Li, "An iterative framework for self-supervised deep speaker representation learning".

[b] Han, Chen, and Qian, "Self-Supervised Learning with Cluster-Aware-DINO for High-Performance Robust Speaker Verification".

[c] Tao et al., "Self-supervised speaker recognition with loss-gated learning".

Outlines

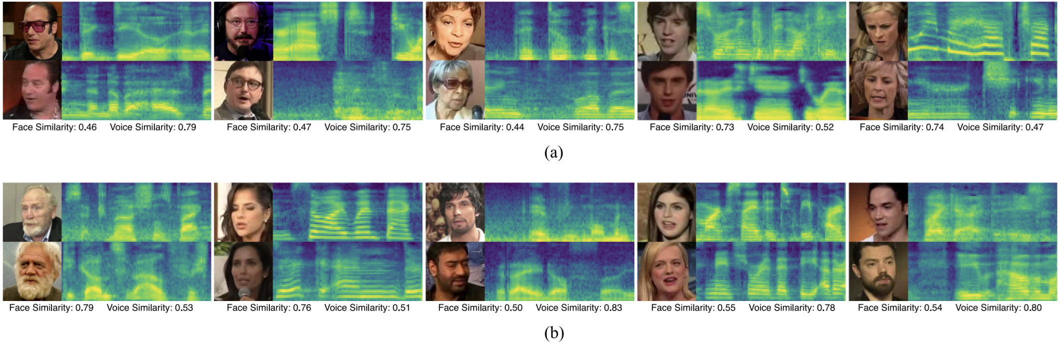


Figure: The speaker similarity based on the audio or visual information[7]

- The upper part shows the speaker's similarity to the same person
- The bottom part shows the speaker's similarity between different persons

[7] Qian, Chen, and Wang, "Audio-visual deep neural network for robust person verification".

Audio-Visual Information Fusion

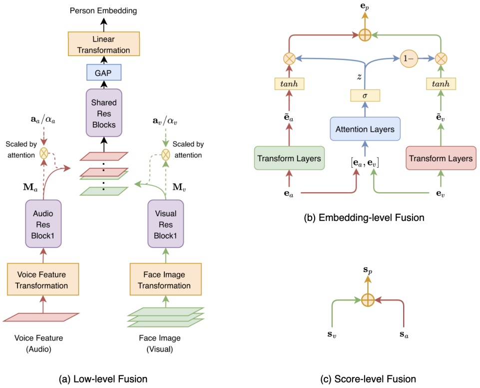


Figure: Audio-visual information fusion at different levels[a]

- Embedding-level fusion performs better than low-level fusion
- The attention mechanism in embedding-level fusion makes it more noise-robust than score-level fusion

[a] Qian, Chen, and Wang, "Audio-visual deep neural network for robust person verification".

Multi-Modal Knowledge Distillation

From audio-visual system to single-modality system

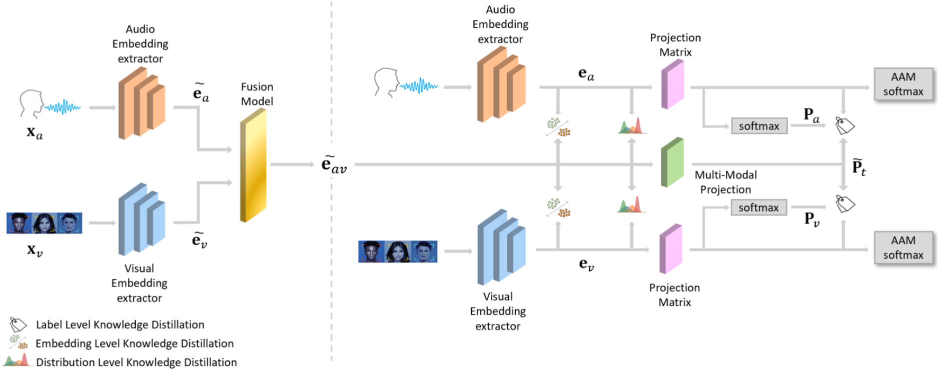


Figure: Knowledge distillation from audio-visual system to single-modality system[8]

[8] Zhang, Chen, and Qian, "Knowledge Distillation from Multi-Modality to Single-Modality for Person Verification".

Multi-Modal Knowledge Distillation

From visual system to audio system

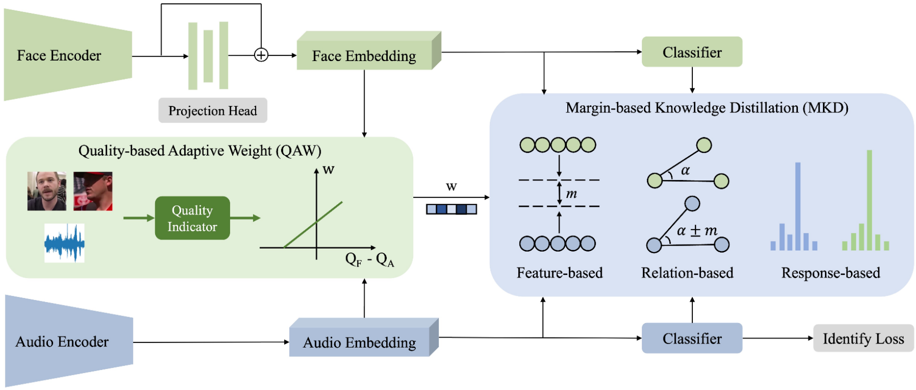


Figure: Knowledge distillation from visual system to audio system[9]

[9] Jin et al., "Cross-modal distillation for speaker recognition".

Outlines

Model Efficiency

In speaker representation learning, we mainly optimize the efficiency of the model from two perspectives: computational efficiency and memory efficiency

- Computation Efficiency
  - Knowledge Distillation
  - Network Quantization
  - Efficient Architecture Design
- Memory Efficiency
  - Reversible Neural Networks

Knowledge Distillation on Speaker Verification Task

- Knowledge distillation from teacher model to student model[a]
- Self-knowledge distillation via feature enhancement[b]
- Knowledge distillation from multi-modality to single-modality[c]

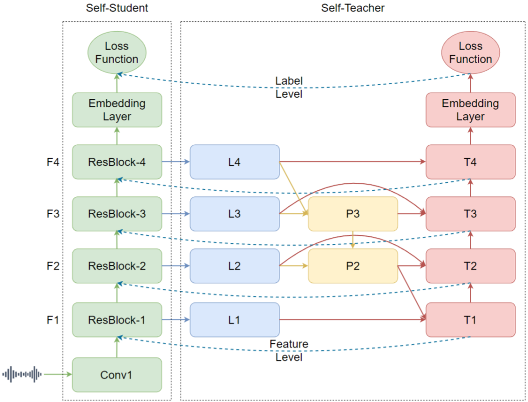


Figure: Schematic diagram of self-knowledge distillation via feature enhancement

[a] Wang et al., "Knowledge Distillation for Small Foot-print Deep Speaker Embedding".

[b] Liu et al., "Self-Knowledge Distillation via Feature Enhancement for Speaker Verification".

[c] Zhang, Chen, and Qian, "Knowledge Distillation from Multi-Modality to Single-Modality for Person Verification".

Quantization achieves model compression by reducing the parameter precision

- Binary Neural Network[a]
- Linear and PoT(Power of Two) quantization[b]
- K-Means based quantization[c]
- Static and adaptive quantizer for binary quantization[d]

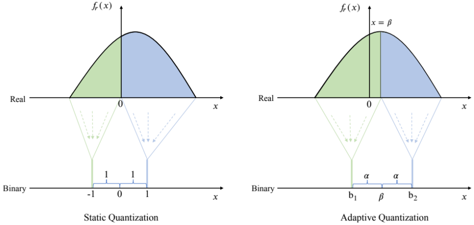


Figure: The overview of static and adaptive binary quantization

[a] Zhu, Qin, and Li, "Binary Neural Network for Speaker Verification".

[b] Liu et al., "Self-Knowledge Distillation via Feature Enhancement for Speaker Verification".

[c] Wang et al., "Adaptive Neural Network Quantization For Lightweight Speaker Verification".

[d] Liu, Wang, and Qian, "Extremely Low Bit Quantization for Mobile Speaker Verification Systems Under 1MB Memory".

Effeicient Architecture Design on Speaker Verification Task

- Depth-First Neural Architecture with Attentive Feature Fusion[a]
- CS-CTCSCONV1D[b](Channel Split Time-Channel-Time Separable 1-dimensional Convolution)
- Asymmetric Enroll-Verify Structure(ECAPA-TDNNLite[c])

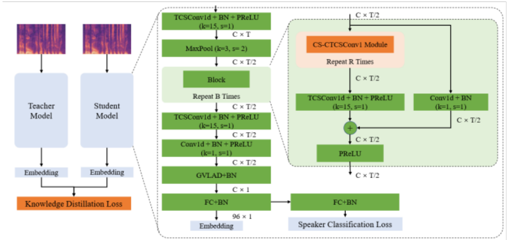


Figure: Schematic of CS-CTCSCONV1D

[a] Liu, Chen, and Qian, "Depth-First Neural Architecture With Attentive Feature Fusion for Efficient Speaker Verification".

[b] Cai et al., "CS-CTCSCONV1D: Small footprint speaker verification with channel split time-channel-time separable 1-dimensional convolution".

[c] Li et al., "Towards Lightweight Applications: Asymmetric Enroll-Verify Structure for Speaker Verification".

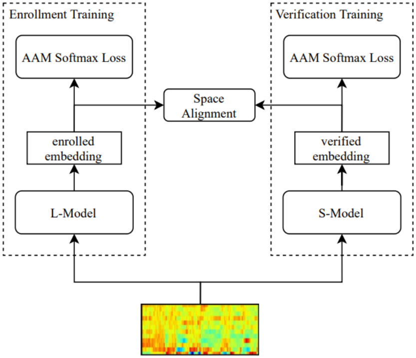


Figure: The training process of the asymmetric structure. Frame-wise input features are fed into the large-scale model and the small-scale model, respectively

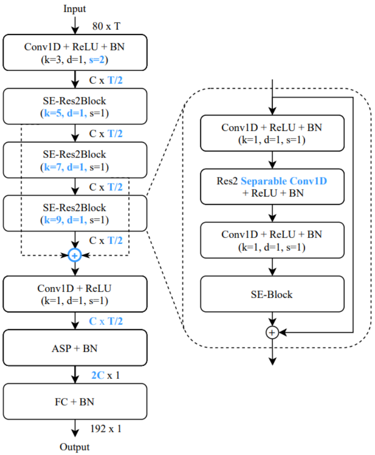


Figure: Schematic of ECAPA-TDNNLite

Table: The experiment results of compressed/quantized ResNet34 and other full-precision compact architectures.

Model	Size (MB)	Bit-width (bit)	Vox1-O EER(%)
KMQAT-ResNet34[10]	3.45	4	**0.957**
PoT-ResNet34[11]	3.45	4	1.09
TWN-ResNet34[12](our impl.)	1.80	2	1.473
b-vector(adaptive)[13]	0.97	1	1.72
ResNet34(binary)[14]	**0.66**	1	5.355
CS-CTCSConv1d	0.96	32	2.62
ECAPA-TDNNLite	1.2	32	3.07

[10] Wang et al., "Adaptive Neural Network Quantization For Lightweight Speaker Verification".

[11] Li et al., "Model Compression for DNN-based Speaker Verification Using Weight Quantization".

[12] Li, Zhang, and Liu, "Ternary weight networks".

[13] Liu, Wang, and Qian, "Extremely Low Bit Quantization for Mobile Speaker Verification Systems Under 1MB Memory".

[14] Zhu, Qin, and Li, "Binary Neural Network for Speaker Verification".

Reversible Neural Networks[a] (RevNets) alleviate the need to store activations in memory during back-propagation. Consequently, RevNets require nearly constant memory costs as the network depth increases.

- Partially reversible networks
- Fully reversible networks

[a] Liu and Qian, "Reversible Neural Networks for Memory-Efficient Speaker Verification".

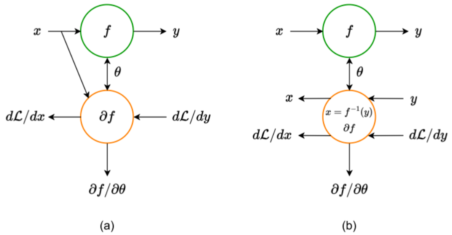


Figure: Comparison between non-reversible operator (a) and reversible operator (b)

Model Memory Efficiency

GPU Memory Usage vs. Parameter Number

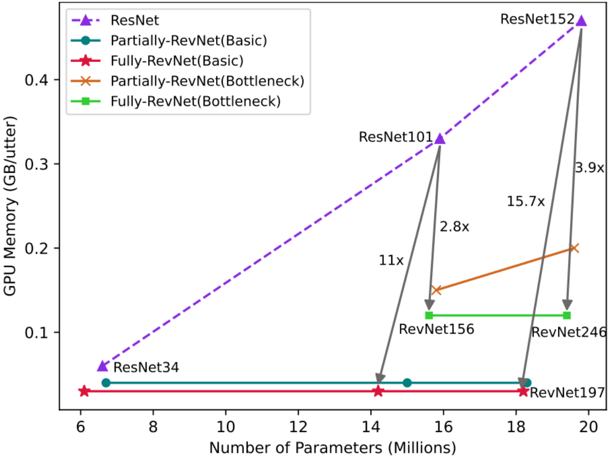


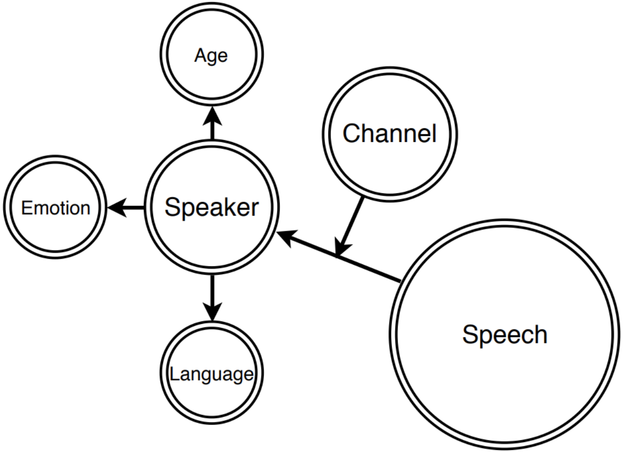
Figure: GPU Memory Usage vs. Parameter Number

Other Work on Model Efficiency

- Thin-ResNet[15]
- Fast-ResNet[16]
- ADMM[17]
- Small Footprint Text-Independent Speaker Verification[18]

[15] Cai, Chen, and Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system".
[16] Chung et al., "In Defence of Metric Learning for Speaker Recognition".
[17] Xu et al., "Mixed Precision Low-Bit Quantization of Neural Network Language Models for Speech Recognition".
[18] Balian et al., "Small footprint text-independent speaker verification for embedded systems".

Robustness in Speaker Representation Learning



The recording environment also introduces variability in modeling speaker identity, influenced by factors like the recording device and microphone distance. To enhance model robustness across different devices, various domain adaptation methods are applied in speaker recognition, including

- Discrepancy-based alignment
- Adversarial learning
- Domain-specific adapter

Discrepancy-based alignment aims to minimize domain discrepancy in a latent feature space and facilitate learning domain-invariant representations. To achieve this goal, choosing a proper divergence measure is at the core of these methods. Widely used measures include MMD[19], correlation alignment (CORAL)[20], etc.

$$\mathcal{L}_{\mathrm{mmd}} \triangleq \sup_{\phi \in \Phi} \left(\mathrm{E}_S \left[\phi \left(S \right) \right] - \mathrm{E}_T \left[\phi \left(T \right) \right] \right) \tag{5}$$

[19] Li, Han, and Song, "CDMA: Cross-Domain Distance Metric Adaptation for Speaker Verification".

[20] Li, Zhang, and Chen, "The coral++ algorithm for unsupervised domain adaptation of speaker recognition".

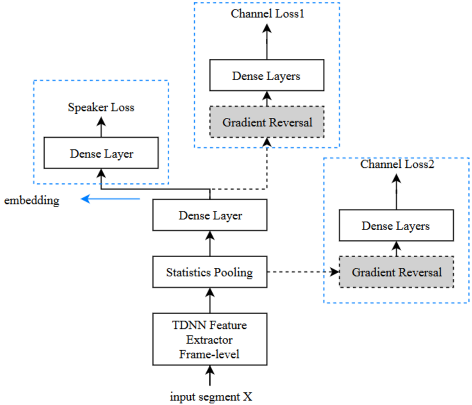
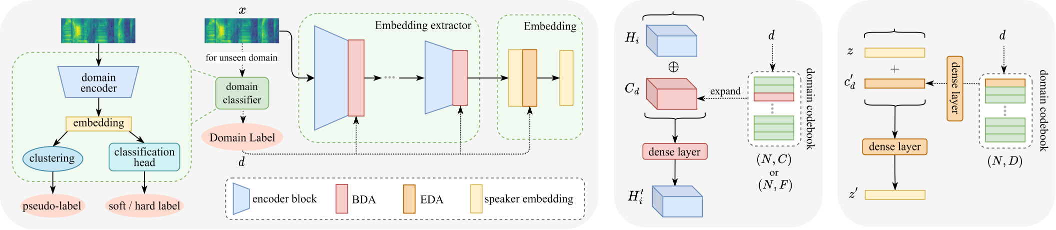


Figure: Structure of channel-level adversarial learning[a]

Adversarial learning employs a domain classifier to eliminate discriminative domain information from features. Min-max optimization in domain-adversarial training minimizes the domain gap and enforces domain-invariant feature extraction[a].

[a] Chen et al., "Channel invariant speaker embedding learning with joint multi-task and adversarial training".

Instead of directly aligning domains with discrepancy measures, incorporating additional modules like domain-specific adapters helps capture and mitigate domain variances, resulting in domain-invariant embeddings.



(a) Model architecture (b) BDA (c) EDA

Figure: Framework with domain-specific adapters[21]

[21] Huang et al., "Enhancing Cross-Domain Speaker Verification through Multi-Level Domain Adapters".

Observation: In real-world scenarios, speaker verification systems may degrade when training on one language and test it on another.

Model Robustness to language mismatch

Language mismatch between enroll / test

Over 40% of the world's population is bilingual, this mismatch happens when the languages used are different for enrollment and test.

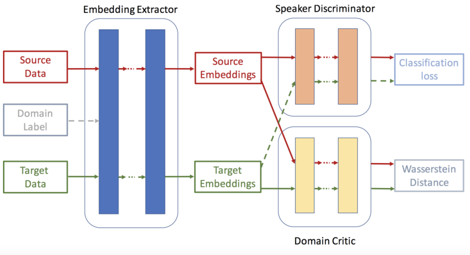


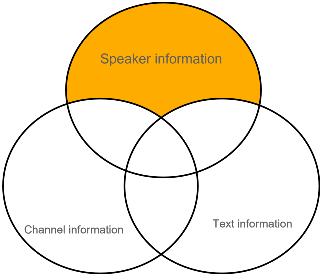
Figure: Structure of language-mismatch adversarial learning

Adversarial learning employs a language classifier to eliminate discriminative language information from features. Min-max optimization in domain-adversarial training minimizes the language gap and enforces language-invariant feature extraction[a][b].

[a] Rohdin et al., "Speaker verification using end-to-end adversarial language adaptation".

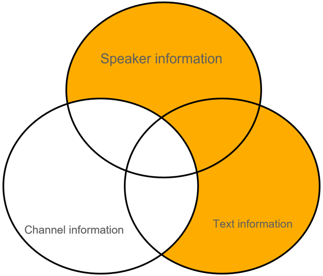
[b] Xia, Huang, and Hansen, "Cross-lingual text-independent speaker verification using unsupervised adversarial discriminative domain adaptation".

Besides the speaker information, the text or content is the most crucial information conveyed through speech.



For text-independent speaker tasks, we only need speaker information

Enroll: Hey Siri; Test: whatever to say



For text-dependent speaker tasks, we also need content information

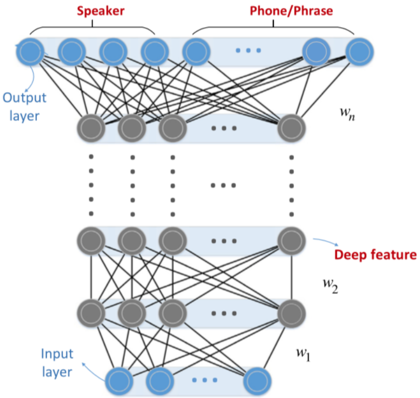
Enroll: Hey Siri; Test: Hey Siri

The representation of **Content Information**

- Phoneme index
- Phoneme posteriors predicted by ASR
- Hidden layer outputs from the ASR Model
- Phrase number (Fix-phrase datasets)
- Normalized phoneme distribution

Text Robustness

multi-task learning in the d-vector framework[22]



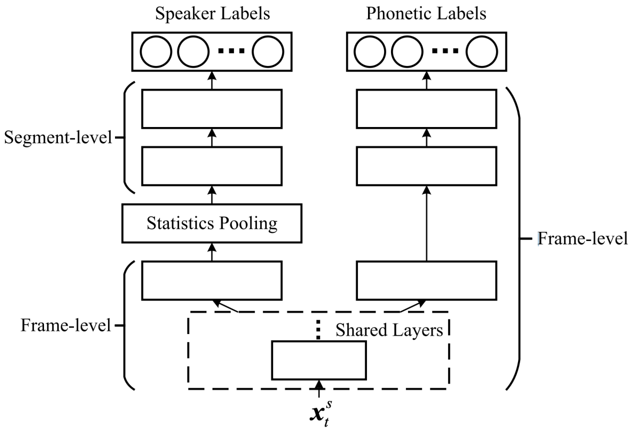
- **Text-dependent** task
- Multi-task at the **frame-level**
- Performance improved

Explicitly modeling phonetic information helps the text-dependent speaker verification task, which is intuitive.

[22] Liu, Yuan, et al. "Deep feature for text-dependent speaker verification." Speech Communication 73 (2015): 1-13.

Text Robustness

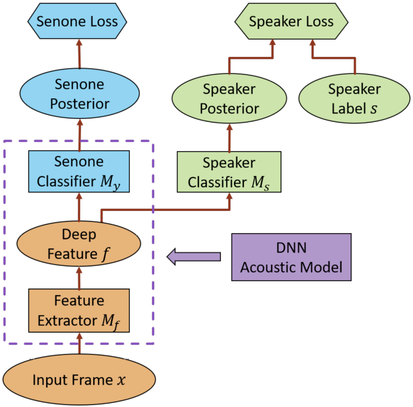
multi-task learning in the x-vector framework[23]



- **Text-independent** task
- Multi-task at the **frame-level**
- Performance improved!

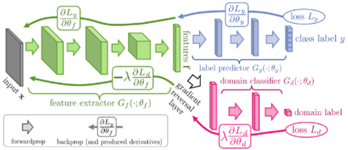
[23] Liu, Yi, et al. "Speaker Embedding Extraction with Phonetic Information." Proc. Interspeech 2018 (2018): 2247-2251.

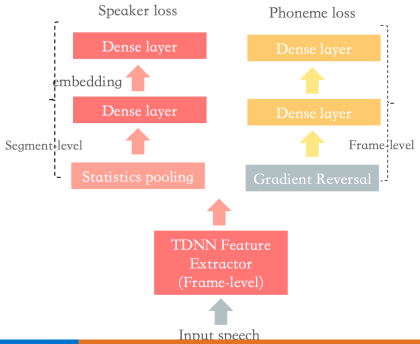
Speaker invariant training for ASR [24]



- Acoustic modelling
- Adversarial training suppressing the speaker effect
- Performance improved

[24] Meng, Zhong, et al. "Speaker-invariant training via adversarial learning." 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018.





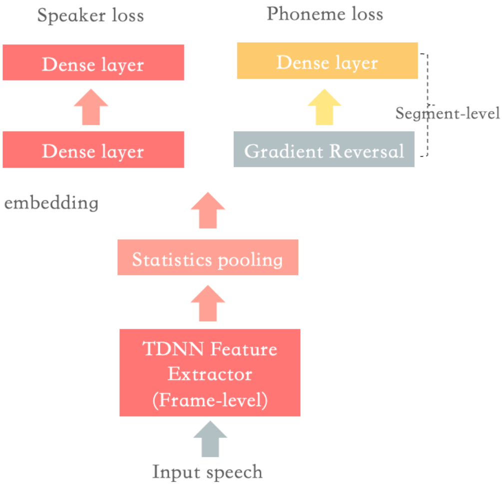
$$\mathcal{L}_s = \mathrm{CE}(M_s(M_f(\mathbf{X})), \mathbf{y}^s)$$

$$\mathcal{L}_p = \frac{1}{N}\sum_{i=1}^{N} \mathrm{CE}(M_p(M_f(\mathbf{x}_i)), \mathbf{y}_i^p)$$

$$\mathcal{L}_{total} = \mathcal{L}_s + \mathcal{L}_p$$

Systems	voxceleb1_O	voxceleb1_E	voxceleb1_H
x-vector baseline	2.361	2.470	4.260
FRM-MT	2.165	2.198	3.911
FRM-ADV	3.143	3.214	5.419

Frame-level multi-task / adversarial training



$$\mathcal{L}_s = \mathsf{CE}(M_s(M_f(\mathbf{X})), \mathbf{y}^s)$$
$$\mathcal{L}_p = \mathsf{CE}(M_p(M_f(\mathbf{x}_i)), \mathbf{y}^p)$$
$$\mathcal{L}_{total} = \mathcal{L}_s + \mathcal{L}_p$$

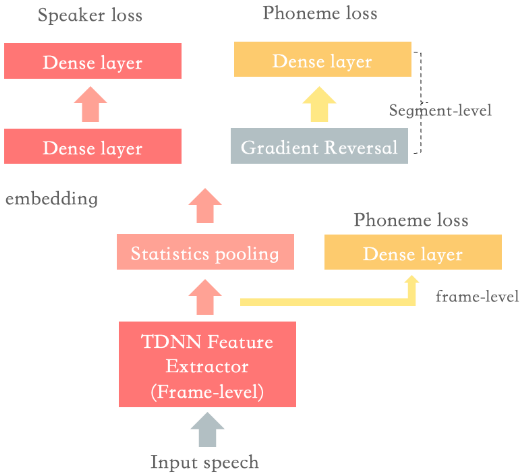
For a given segment $\mathbf{x}$ with N frames, segment-level phoneme label $\mathbf{y}^p$ is

$$\mathbf{y}^p = \{y_1, y_2, \ldots, y_C\}$$
$$y_c = \frac{N_c}{N}$$

where C is the size of the chosen phoneme set. N_c denotes the number of occurrences of the c-th phoneme in $\mathbf{x}$

Text Robustness

Frame-level multitask + segment-level adversarial learning



Systems	voxceleb1_O	voxceleb1_E	voxceleb1_H
x-vector baseline	2.361	2.470	4.260
SEG-MT	2.175	2.330	4.059
SEG-ADV	2.154	2.198	3.923

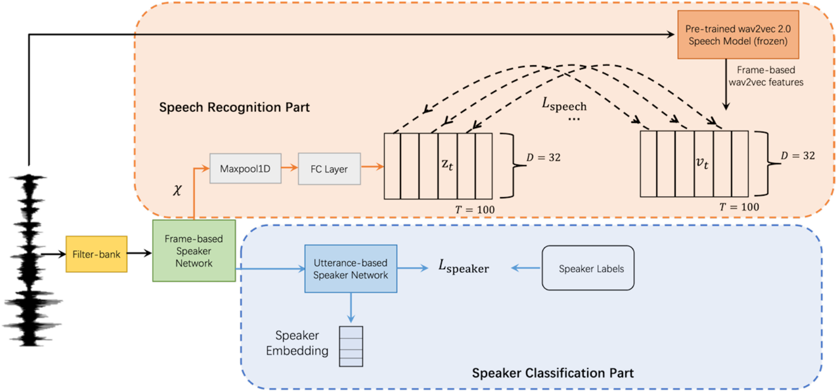
Segment-level Multitask/Adversarial training

Systems	voxceleb1_O	voxceleb1_E	voxceleb1_H
x-vector baseline	2.361	2.470	4.260
FRM-MT	2.165	2.198	3.911
SEG-ADV	2.154	2.198	3.923
COMBINE	2.013	2.030	3.819

Frame-level multi-task + segment-level adv training

Text Robustness

Multi-task training with high-level content representation[25]



[25] Jin, Tu, and Mak, "Phonetic-aware speaker embedding for far-field speaker verification".

Extract phonetic bottleneck (PBN) from a pretrained ASR model and combine it with the filterbanks[26]

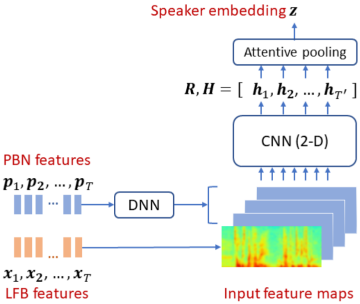


Fig. 1: *Implicit phonetic attention by combining LFB and PBN features at the input layer (LFB: log filter bank; PBN: phonetic bottleneck).*

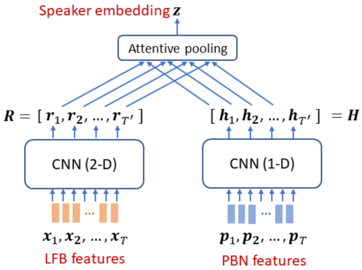
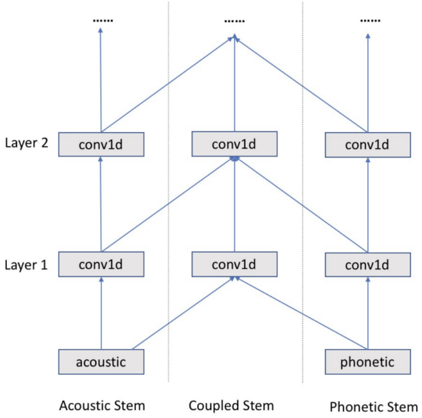


Fig. 2: *Explicit phonetic attention by routing LFB and PBN features through separate networks (LFB: log filter bank; PBN: phonetic bottleneck).*

[26] Zhou T, Zhao Y, Li J, et al. CNN with phonetic attention for text-independent speaker verification, ASRU

Text Robustness

Phoneme-aware speaker embedding learning



27

Table 1: *Network configurations of PacNet*

Layer 7	Linear	In=1024 Out=1000		
Layer 6	Pooling	In=1024 Out=1024		
Layer 5	Conv1d	In=2048 Out=1024		
Layer 4	Conv1d kernel=5	Out=512 In=512	Out=1024 In=2048	Out=512 In=512
Layer 3	Conv1d kernel=5	Out=512 In=512	Out=1024 In=2048	Out=512 In=512
Layer 2	Conv1d kernel=5	Out=512 In=512	Out=1024 In=2048	Out=512 In=512
Layer 1	Conv1d kernel=5	Out=512 In=40	Out=1024 In=140	Out=512 In=100
Stem		Acoustic	Coupled	Phonetic

- Triplet loss instead of softmax loss

[27] Zheng, Lei, and Suo, "Phonetically-Aware Coupled Network For Short Duration Text-Independent Speaker Verification."

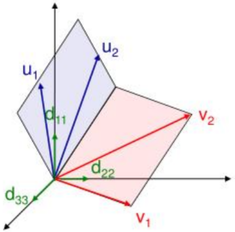
The general idea: **Decomposition and Reconstruction. The application is far more than speaker modeling**

Applications

- Speaker representation learning
- Voice conversion
- Speech synthesis/ Voice Cloning

Joint Factor Analysis for speaker representation learning[28]

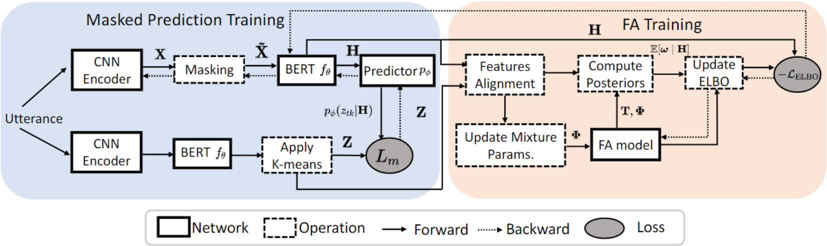
$$\mathbf{M} = \mathbf{M}^{\text{UBM}} + \mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z} + \mathbf{U}\mathbf{x}$$

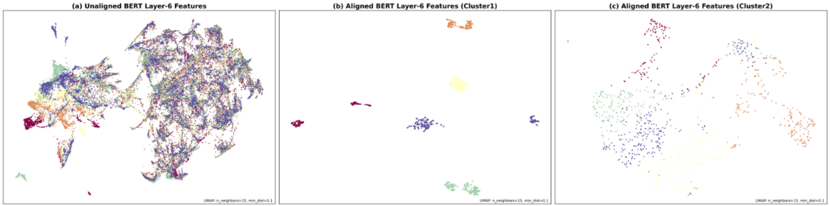


- Gaussian priors assumed for factors $\mathbf{y}$, $\mathbf{z}$, $\mathbf{x}$
- $\mathbf{M}^{\text{UBM}}$, $\mathbf{V}$, $\mathbf{D}$, $\mathbf{U}$ are estimated using EM algorithm
- $\mathbf{V}$ captures main speaker variability (Eigen voices)
- $\mathbf{D}$ captures channel variability
- $\mathbf{U}$ captures residual variability

[28] Kenny, Patrick, et al. "Joint factor analysis versus eigenchannels in speaker recognition." TASLP 2007

Neural Factor Analysis: Neglect the phoneme variations by additional alignment [29]

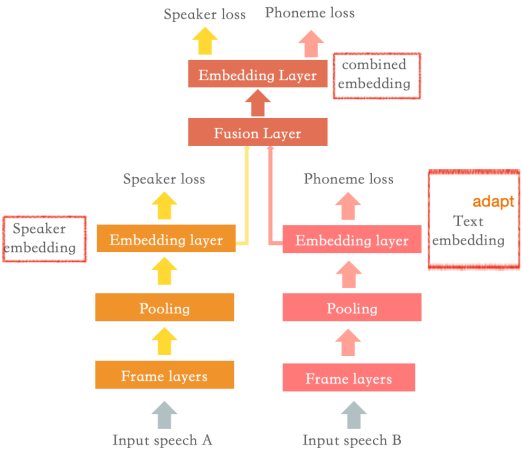




[29] Lin W W, He C H, .et, Self-supervised Neural Factor Analysis for Disentangling Utterance-level Speech Representations, ICML 2023

Speech representation disentanglement

Decouple and Reorganization of Phonetic Information[30]



- Segment-level reconstruction
- Decoupling the speaker and text information
- For the text-independent task, we neglect the text information
- For the text-dependent task, we use the combined embedding
- Text-adaptive task: Modify the text information in the embedding while keeping the speaker identity. (Change the enrollment keyword)

[30] Yang Y*, Wang S*, Gong X, et al. Text adaptation for speaker verification with speaker-text factorized embeddings. ICASSP 2020

Decouple and Reorganization of Phonetic Information[31]

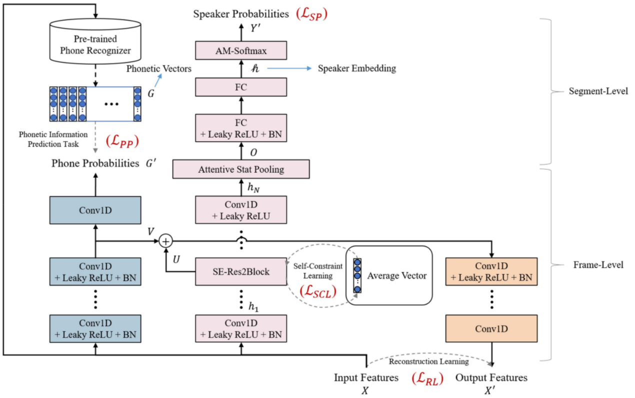
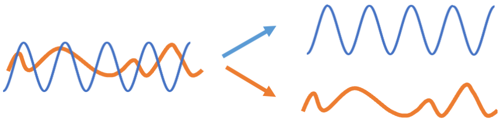


Fig. 3. The architecture of the proposed DROP-TDNN x-vector system. DROP-TDNN consists of three training procedures, including phonetic information prediction, reconstruction and speaker recognition.

- Frame-level Reconstruction
- Center frame-level speaker representations towards its mean
- Coarse-grained phoneme categories (Vowel, semi-vowel, affricate, ...)

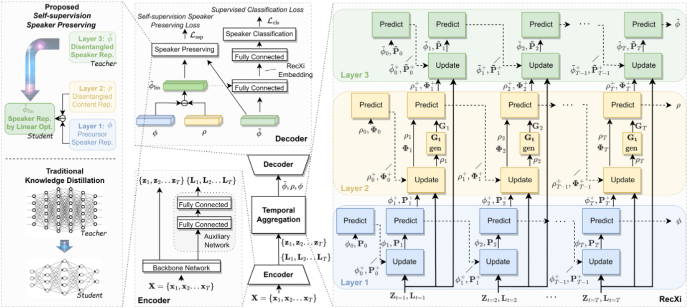
[31] Hong Q B, Wu C H, Wang H M. Decomposition and Reorganization of Phonetic Information for Speaker Embedding Learning. TASLP 2023

Disentangle **Static** and **Dynamic** components in Speech



by three **Gaussian inference layers**

and a novel **speaker preserving self-supervision**



[32] Liu T C, Disentangling Voice and Content with Self-Supervision for Speaker Recognition

Ensure the first layer representations contain content-related information, the subsequent residual layers will naturally fill in the gaps with remaining details—specifically, modeling the paralinguistic information.[33]

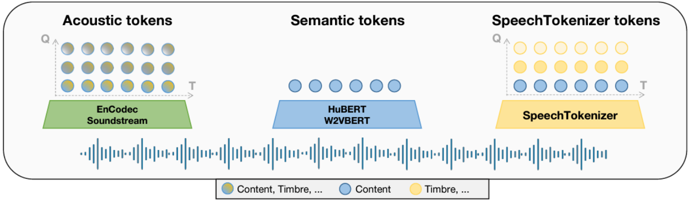


Figure 1: Illustration of information composition of different discrete speech representations. Speech tokens are represented as colored circles and different colors represent different information.

[33] Zhang X, Zhang D, Li S, et al. SpeechTokenizer: Unified Speech Tokenizer for Speech Large Language Models[J]. arXiv preprint arXiv:2308.16692, 2023.

Semantic Distillation to enable the disentanglement

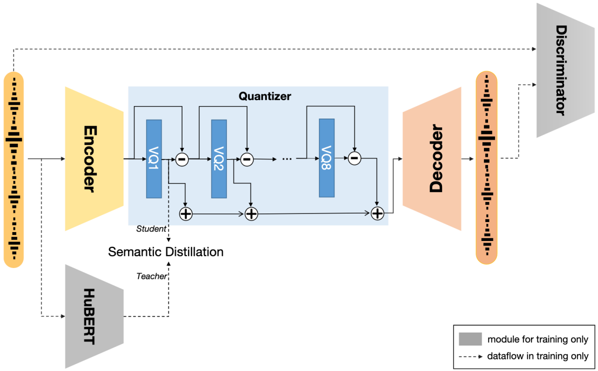


Figure 2: Illustration of SpeechTokenizer framework.

- **Continious distillation** Output of hubert's 9-th layer/Average across all layers
  $\mathcal{L}_{\text{distll}} = \frac{1}{T}\sum_{t=1}^{T} \log \sigma \left(\cos\left(A q_1^t, s^t\right)\right)$
- **Discrete distillation** pseudo-label prediction
  $\mathcal{L}_{\text{distll}} = -\frac{1}{T}\sum_{t=1}^{T} u^t \log\left(\text{Softmax}\left(A q_1^t\right)\right)$

Assume hubert is a perfect semantic encoder

Outlines

Explore model capacity via probing tasks

Analyze the information encoded

Assumption: If a certain attribute is encoded in the speaker representation, the accuracy of a classifier predicting this property depends on how well it's embedded.[34,35,36,37]

- Speaker-related attributes: identity, gender, and speaking rate.
- Text-related factors: spoken terms, word order, and utterance length.
- Channel-related elements include the handset ID and noise type.

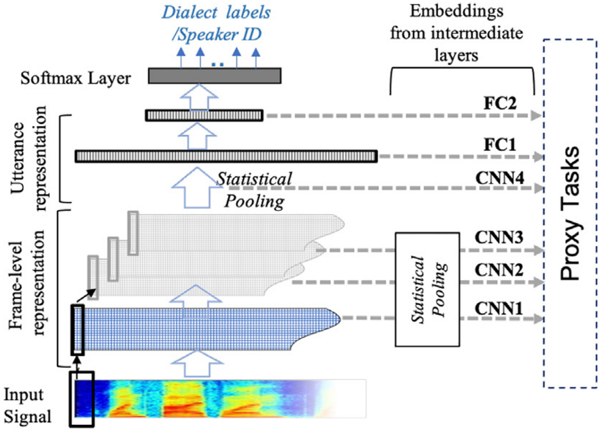
[34] Wang, Qian, and Yu, "What does the speaker embedding encode?"

[35] Belinkov and Glass, "Analyzing hidden representations in end-to-end automatic speech recognition systems".

[36] Raj et al., "Probing the information encoded in x-vectors".

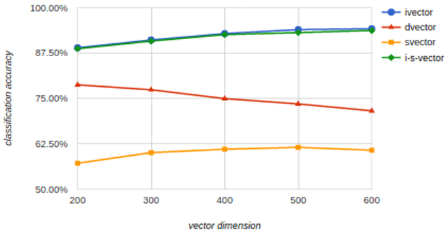
[37] Zhao et al., "Probing Deep Speaker Embeddings for Speaker-related Tasks".

Illustration of the paradigm: Probing pretrained embeddings with proxy tasks[38]

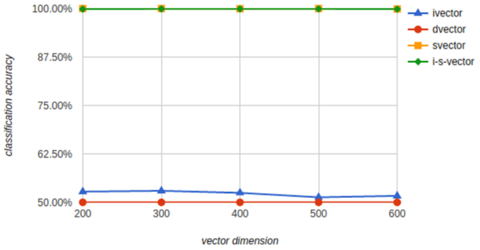


[38] Chowdhury, Durrani, and Ali, "What do end-to-end speech models learn about speaker, language and channel information? a layer-wise and neuron-level analysis".

Examples of the speaker identity task and word order task[39]



Speaker identity task



Word order task

[39] Wang, Qian, and Yu, "What does the speaker embedding encode?"

In the context of speaker modeling, f is the speaker classifier, c represents the class, θ represents the trainable model parameters.

$$y^c = f_c(x;\theta)$$

For the k-th activation map A^k(e.g. k represents the k-th channel), each entry w_{ij}^{kc} is defined as

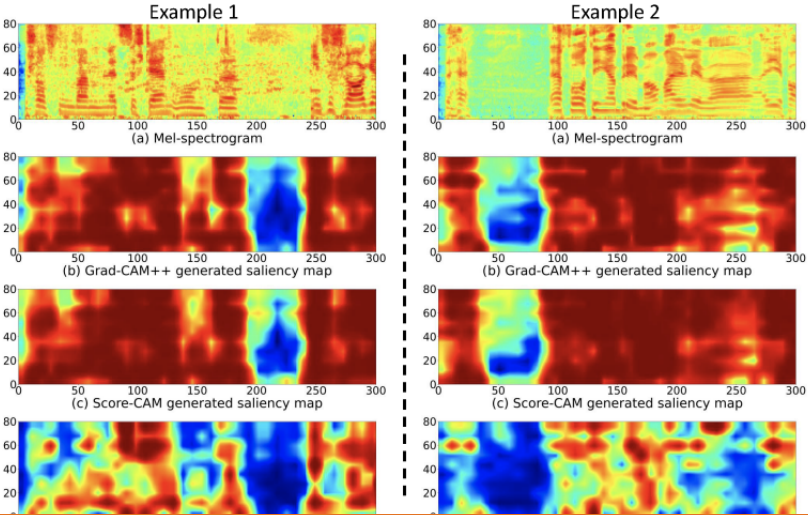
$$w_{ij}^{kc} = \mathrm{ReLU}\left(\frac{\partial y^c}{\partial A_{ij}^k}\right)$$

Saliency map is defined as the linear combination

$$S_{ij}^c = \mathrm{ReLU}\left(\sum_k w_{ij}^{kc} \cdot A_{ij}^k\right)$$

Measuring the importance through visualization

Visualization in Speaker Recognition[40, 41]



Outlines

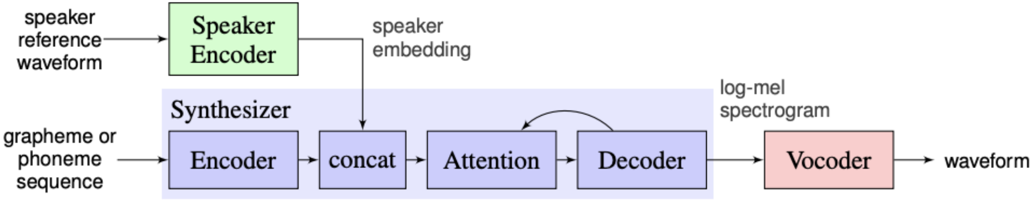
New paradigm to model speakers

Different tasks, different approaches

1. Pretrained speaker embeddings as additional inputs
2. Joint training to learn task-specific embeddings
3. Implict speaker modeling

New paradigm to model speakers

Example: Explict speaker modeling for Zero-shot TTS[42],[43],[44]



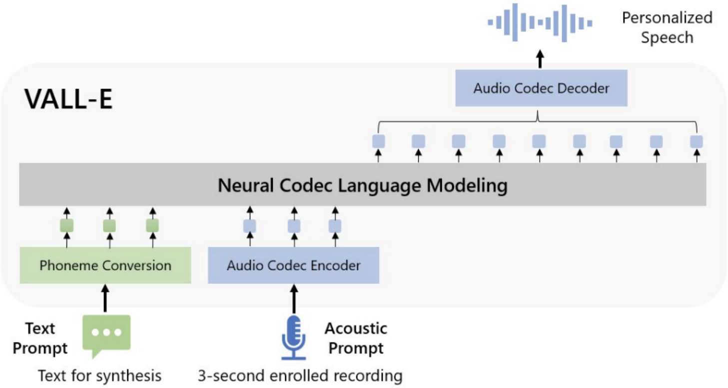
[42] Jia et al., "Transfer learning from speaker verification to multispeaker text-to-speech synthesis".

[43] Casanova et al., "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone".

[44] Wu et al., "Adaspeech 4: Adaptive text to speech in zero-shot scenarios".

New paradigm to model speakers

Example: Implict speaker modeling for Zero-shot TTS[45],[46],[47]

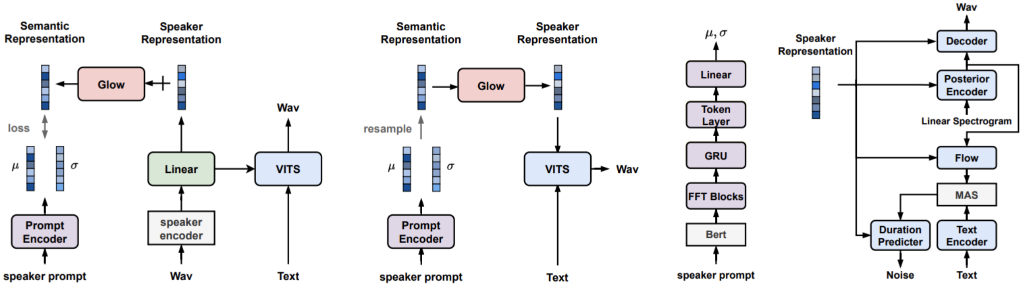


[45] Wang et al., "Neural codec language models are zero-shot text to speech synthesizers".

[46] Du et al., "UniCATS: A Unified Context-Aware Text-to-Speech Framework with Contextual VQ-Diffusion and Vocoding".

[47] Le et al., "Voicebox: Text-guided multilingual universal speech generation at scale".

Example: towards controlability and new voice generation[48],[49],[50],[51]



(a) training process (b) inference process (c) prompt encoder (d) zero-shot VITS

[48] Zhang et al., "PromptSpeaker: Speaker Generation Based on Text Descriptions".

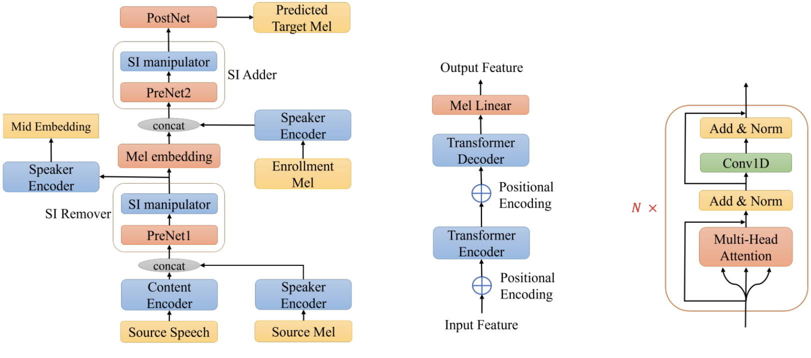
[49] Stanton et al., "Speaker generation".

[50] Shimizu et al., "PromptTTS++: Controlling Speaker Identity in Prompt-Based Text-to-Speech Using Natural Language Descriptions".

[51] Bilinski et al., "Creating new voices using normalizing flows".

New paradigm to model speakers

Example: Explict speaker modeling for zero-shot voice conversion[52][53][54]



(a) Proposed System (b) SI Manipulator (c) Encoder and decoder architecture

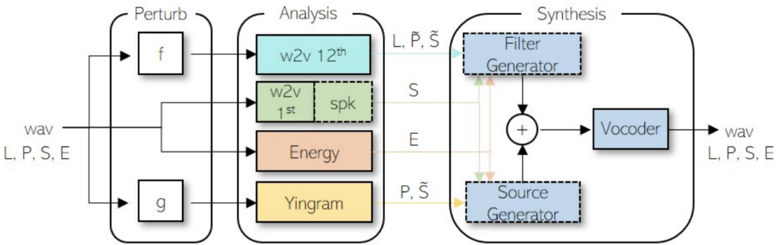
[52] Zhang et al., "SIG-VC: A Speaker Information Guided Zero-Shot Voice Conversion System for Both Human Beings and Machines".

[53] Chen and Duan, "ControlVC: Zero-Shot Voice Conversion with Time-Varying Controls on Pitch and Rhythm".

[54] Hussain et al., "ACE-VC: Adaptive and Controllable Voice Conversion Using Explicitly Disentangled Self-Supervised Speech Representations".

New paradigm to model speakers

Example: Implict speaker modeling for zero-shot voice conversion[55],[56],[57]



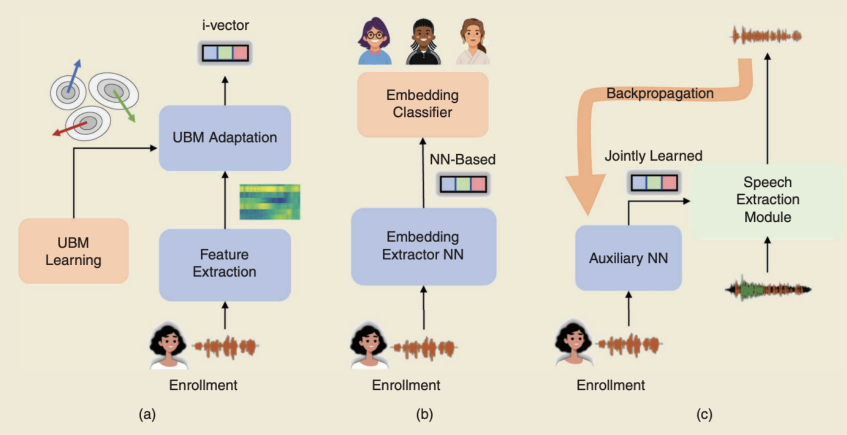
[55] Choi et al., "Neural analysis and synthesis: Reconstructing speech from self-supervised representations".

[56] Wu and Lee, "One-shot voice conversion by vector quantization".

[57] Wu, Chen, and Lee, "Vqvc+: One-shot voice conversion by vector quantization and u-net architecture".

New paradigm to model speakers

Example: Explict speaker modeling for target speaker extraction[58][59][60]



[58] Zmolikova et al., "Neural Target Speech Extraction: An overview".

[59] Delcroix et al., "Single channel target speaker extraction and recognition with speaker beam".

[60] Ge et al., "Spex+: A complete time domain speaker extraction network".

New paradigm to model speakers

Example: Implict speaker modeling for target speaker extraction[61],[62]

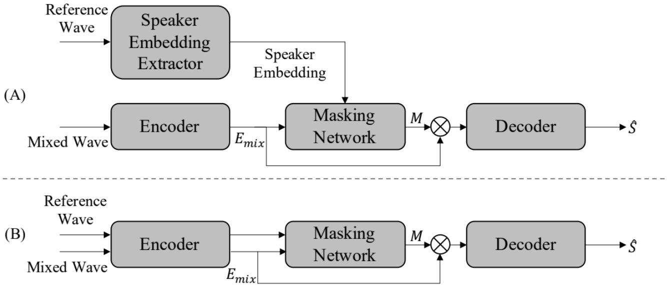


Figure 1: *(A) is the diagram of a typical time-domain target speaker extraction method. (B) is the diagram of our proposed method.* $\otimes$ *is an operation for element-wise product.*

[61] Zeng et al., "SEF-Net: Speaker Embedding Free Target Spekaer Extraction Network".

[62] Yang et al., "Target Speaker Extraction with Ultra-Short Reference Speech by VE-VE Framework".

Outlines

Wespeaker is a speaker embedding learning toolkit designed for **research** and **production** purposes, it characterized by

- Lightwight codebase
- SOTA performance
- Discriminative and SSL based paradigms
- Runtime/Deployment support
- Adopted by research groups from both companies and academic institutions:
  - Tencent
  - Meituan
  - China Telecom
  - NVIDIA
  - Tsinghua University
  - The Chinese University of Hong Kong (Shenzhen)
  - Shanghai Jiao Tong University
  - University of Science and Technology of China
  - National University of Singapore
  - Institute for Infocomm Research (I2R)
  - Brno University of Technology.

- Data Preparation
  - Data Downloading
  - Formating
  - Transformation
- Model Training
  - On-the-fly Data Augmentation
  - Model Selection
  - Large-margin Fine-tuning
- Backend Scoring
  - As-norm
  - PLDA / A-PLDA
  - Score Calibration (coming soon)

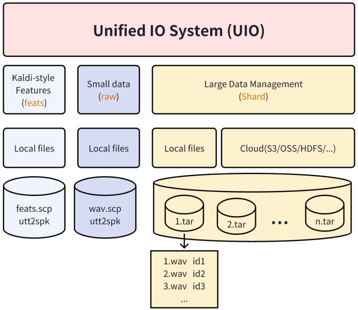


Figure: Unified I/O system

Unified I/O system

- Also adopted in wenet ASR toolkit
- Inspired by webdataset and tfrecord

Idea

- Raw: load wav and label files from disk (small data)
- Shard:
  - Pack a set of small files into a bigger shard
  - Read and decompress the shard files on-the-fly
- Feat: Compatible with kaldi-style feature files
- Effectively loading large-scale datasets

Step 1: Download and prepare metadata

```
if [ ${stage} -le 1 ] && [ ${stop_stage} -ge 1 ]; then
  echo "Prepare datasets ..."
  ./local/prepare_data.sh --stage 2 --stop_stage 4 --data ${data}
fi
```

Step 2: Covert train and test data

```
if [ ${stage} -le 2 ] && [ ${stop_stage} -ge 2 ]; then
  echo "Covert train and test data to ${data_type}..."
  for dset in vox2_dev vox1; do
    if [ $data_type == "shard" ]; then
      python tools/make_shard_list.py --num_utts_per_shard 1000 \
          --num_threads 16 \
          --prefix shards \
          --shuffle \
          ${data}/$dset/wav.scp ${data}/$dset/utt2spk \
          ${data}/$dset/shards ${data}/$dset/shard.list
    else
      python tools/make_raw_list.py ${data}/$dset/wav.scp \
          ${data}/$dset/utt2spk ${data}/$dset/raw.list
    fi
  done
fi
```

Step3: Start training

```
if [ ${stage} -le 3 ] && [ ${stop_stage} -ge 3
    ]; then
  echo "Start training ..."
  num_gpus=$(echo $gpus | awk -F ',' '{print NF
      }')
  torchrun --standalone --nnodes=1 --
      nproc_per_node=$num_gpus \
    wespeaker/bin/train.py --config $config \
      --exp_dir ${exp_dir} \
      --gpus $gpus \
      --num_avg ${num_avg} \
      --data_type "${data_type}" \
      --train_data ${data}/vox2_dev/${data_type
          }.list \
      --train_label ${data}/vox2_dev/utt2spk \
      --reverb_data ${data}/rirs/lmdb \
      --noise_data ${data}/musan/lmdb \
      ${checkpoint:+--checkpoint $checkpoint}
fi
```

Dataset Config:

```
dataset_args:
  speed_perturb: True
  num_frms: 200
  aug_prob: 0.6
  # prob to add reverb
      & noise aug per
      sample
  fbank_args:
    num_mel_bins: 80
    frame_shift: 10
    frame_length: 25
    dither: 1.0
  spec_aug: False
  spec_aug_args:
    num_t_mask: 1
    num_f_mask: 1
    max_t: 10
    max_f: 8
    prob: 0.6
```

Data Augmentation:

```
# add noise
dataset = Processor(
      dataset, processor
      .add_reverb_noise,
       reverb_data,
      noise_data,
      resample_rate,
      aug_prob)
# speed perturb
dataset = Processor(
      dataset, processor
      .speed_perturb,
      len(spk2id_dict))
# specaug
dataset = Processor(
      dataset, processor
      .spec_aug, **
      configs['
      spec_aug_args'])
```

Model Arch:

- ResNet Series
- TDNN
- ECAPA-TDNN
- RepVGG
- CAM++

Pooling Methods:

- TSTP
- ASTP
- MQMHASTP

Loss Function:

- add_margin
- arc_margin
- sphere
- sphereface2
- intertopk
- subcenter

Model Config:

```
model: ResNet34
    # ECAPA, CAMPPlus,
          REPVGG,
          ResNet152
model_args:
  feat_dim: 80
  embed_dim: 256
  pooling_func: "TSTP"
      # TSTP, ASTP,
      MQMHASTP
  two_emb_layer: False
projection_args:
  project_type: "
       arc_margin"
  # add_margin,
       arc_margin,
       sphere,
       sphereface2,
       softmax,
       aam_intertopk
  scale: 32.0
```

Back-end Support:

- Cosine
- PLDA
- Adapt-PLDA

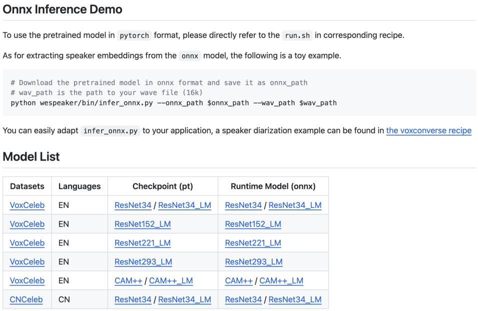
Others:

- Score normalization
- QMF based Calibration

Scoring:

```
if [ ${stage} -le 5 ] && [ ${stop_stage} -ge 5
    ]; then
  echo "Score ..."
  local/score.sh \
    --stage 1 --stop-stage 2 \
    --data ${data} \
    --exp_dir $exp_dir \
    --trials "$trials"
fi

if [ ${stage} -le 6 ] && [ ${stop_stage} -ge 6
    ]; then
  echo "Score norm ..."
  local/score_norm.sh \
    --stage 1 --stop-stage 3 \
    --score_norm_method $score_norm_method \
    --cohort_set vox2_dev \
    --top_n $top_n \
    --data ${data} \
    --exp_dir $exp_dir \
    --trials "$trials"
fi
```



Onnx Inference Demo

To use the pretrained model in `pytorch` format, please directly refer to the `run.sh` in corresponding recipe.

As for extracting speaker embeddings from the `onnx` model, the following is a toy example.

```
# Download the pretrained model in onnx format and save it as onnx_path
# wav_path is the path to your wave file (16k)
python wespeaker/bin/infer_onnx.py --onnx_path $onnx_path --wav_path $wav_path
```

You can easily adapt `infer_onnx.py` to your application, a speaker diarization example can be found in the voxconverse recipe

Model List

Datasets	Languages	Checkpoint (pt)	Runtime Model (onnx)
VoxCeleb	EN	ResNet34 / ResNet34_LM	ResNet34 / ResNet34_LM
VoxCeleb	EN	ResNet152_LM	ResNet152_LM
VoxCeleb	EN	ResNet221_LM	ResNet221_LM
VoxCeleb	EN	ResNet293_LM	ResNet293_LM
VoxCeleb	EN	CAM++ / CAM++_LM	CAM++ / CAM++_LM
CNCeleb	CN	ResNet34 / ResNet34_LM	ResNet34 / ResNet34_LM

Figure: Pretrained Model List

Export Jit:

```
if [ ${stage} -le 7 ] && [ ${stop_stage} -ge 7 ]; then
  echo "Export the best model ..."
  python wespeaker/bin/export_jit.py \
    --config $exp_dir/config.yaml \
    --checkpoint $exp_dir/models/avg_model.pt \
    --output_file $exp_dir/models/final.zip
fi
```

Export Onnx:

```
exp=exp  # Change it to your experiment dir
onnx_dir=onnx
python wespeaker/bin/export_onnx.py \
  --config $exp/config.yaml \
  --checkpoint $exp/avg_model.pt \
  --output_model $onnx_dir/final.onnx
```

Wespeaker

Deployment and product-oriented setups

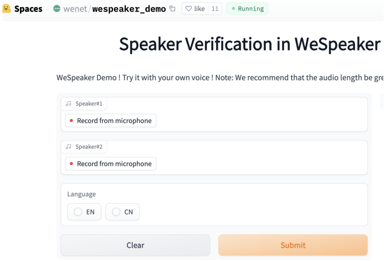


Figure: Wespeaker Demo Page

Command-line usage:

```
wespeaker --task embedding --audio_file audio.wav --output_file embedding.txt -g 0
wespeaker --task embedding_kaldi --wav_scp wav.scp --output_file /path/to/embedding -g 0
wespeaker --task similarity --audio_file audio.wav --audio_file2 audio2.wav -g 0
```

Python programming usage:

```
import wespeaker

model = wespeaker.load_model('chinese')
# set_gpu to enable the cuda inference, number < 0 means using CPU
model.set_gpu(0)
embedding = model.extract_embedding('audio.wav')
utt_names, embeddings = model.extract_embedding_list('wav.scp')
similarity = model.compute_similarity('audio1.wav', 'audio2.wav')
diar_result = model.diarize('audio.wav')
```

Table: 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 EER(%)	voxceleb1_O minDCF	voxceleb1_E EER(%)	voxceleb1_E minDCF	voxceleb1_H EER(%)	voxceleb1_H minDCF
IDLab VoxSRC 2020[63]	ECAPA-TDNN	0.870	0.107	1.120	0.132	2.120	0.210
BUT VoxSRC 2019[64]	ResNet34	1.310	0.154	1.380	0.163	2.500	0.233
AsvSubtools[1]	ECAPA-TDNN	0.856	-	-	-	-	-
	Conformer	0.792	-	-	-	-	-
SpeechBrain[2]	TDNN	3.23	-	-	-	-	-
	ECAPA-TDNN	0.90	-	-	-	-	-
	ECAPA-TDNN *	1.30	-	1.98	-	3.62	-
Nemo[3]	TDNN	1.96	-	-	-	-	-
	ECAPA-TDNN	0.92	-	-	-	-	-
	titanet_large	0.66	-	-	-	-	-
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
	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**

[63] Desplanques, Thienpondt, and Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification".

[64] Zeinali et al., "But system description to voxceleb speaker recognition challenge 2019".

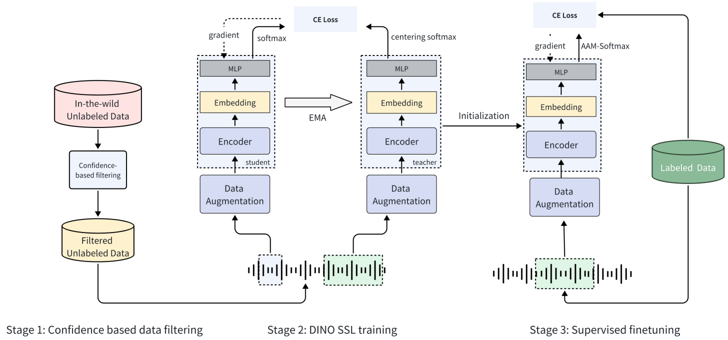
Table: Results on the CNCeleb evaluation set

Toolkits	Architecture	EER(%)	minDCF
ASVSubtools	ResNet34	9.141	0.463
Wespeaker	TDNN	8.960	0.446
	ECAPA-TDNN	7.395	0.372
	CAM++	7.052	0.368
	ResNet34	6.492	0.354
	ResNet221	**5.655**	**0.330**

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

Toolkits	Paradigm	Architecture	VoxCeleb1_O	VoxCeleb1_E	VoxCeleb1_H
3Dspeaker	RDINO	ECAPA-TDNN (C1024)	3.16	-	-
wespeaker	SimCLR	ECAPA-TDNN	8.523	9.417	14.907
	MoCo	ECAPA-TDNN	8.709	9.287	14.756
	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

A comprehensive example of using Wespeaker[65],[66]



Step 1: Clustering-based speaker diarization system, filter out low-quality segments.
Step 2: Train a DINO system on the filtered data
Step 3: Fintuning the pretrained DINO system in a supervised setup.

[65] Wang et al., "Leveraging In-the-Wild Data for Effective Self-Supervised Pretraining in Speaker Recognition".

[66] Yu et al., "AutoPrep: An Automatic Preprocessing Framework for In-the-Wild Speech Data".

Table: Comparison of performance on CNCeleb-Eval with other pretrain-finetune methods.

System	Pretraining Configurations			Finetuning Configurations		EER(%)	MinDCF
	Data	Model	Role	Data	Model		
[67]	VoxCeleb2	ECAPA-TDNN	Init	CNCeleb1	ECAPA-TDNN	10.65	-
[68]	VoxCeleb2	ECAPA-TDNN	Init	CNCeleb1	ECAPA-TDNN	8.710	0.422
[69]	VoxCeleb2	ECAPA-TDNN	Init	CNCeleb1	ECAPA-TDNN	10.03	0.539
[70]	CNCeleb1	HuBERT (94.6M)	Frontend	CNCeleb1	HuBERT + ECAPA-TDNN	10.86	-
[7]	CNCeleb-Train	HuBERT (94.6M)	Frontend	CNCeleb-Train	HuBERT + ECAPA-TDNN	8.890	-
[7]	CNCeleb-Train	Conformer (172.2M)	Frontend	CNCeleb-Train	Conformer + MHFA	7.730	0.406
[71] *	Mix 94k hr	WavLM (94.7M)	Frontend	VoxCeleb2 + CNCeleb-Train	WavLM+MAM+MHFA	6.890	0.378
[72]**	WenetSpeech	Conformer (18.8M)	Init	CNCeleb-Train	Conformer	7.420	0.443
Ours	WenetSpeech	ECAPA-TDNN	Init	CNCeleb1	ECAPA-TDNN	7.373	0.383
Ours	+ filtering	ECAPA-TDNN	Init	CNCeleb1	ECAPA-TDNN	7.339	0.377
Ours	WenetSpeech	ECAPA-TDNN	Init	CNCeleb-Train	ECAPA-TDNN	6.738	0.338
Ours	+ filtering	ECAPA-TDNN	Init	CNCeleb-Train	ECAPA-TDNN	**6.474**	**0.331**

[67] Heo et al., "Self-supervised curriculum learning for speaker verification".
[68] Kang et al., "Augmentation adversarial training for self-supervised speaker representation learning".
[69] Han et al., "Improving dino-based self-supervised speaker verification with progressive cluster-aware training".
[70] Peng et al., "Improving speaker verification with self-pretrained transformer models".
[71] Peng et al., "Parameter-efficient transfer learning of pre-trained Transformer models for speaker verification using adapters".
[72] Liao et al., "Towards a unified conformer structure: from asr to asv task".

Conclusions and Takeaway

- Speaker modeling is not all about speaker recognition.
- Speaker modeling is more than embedding learning.
- Customize the speaker modeling approach for the specific task.
- Try wespeaker! `https://github.com/wenet-e2e/wespeaker`

Contact

Email: wsstriving@gmail.com