IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 1, 2021, accepted June 14, 2021, date of publication June 17, 2021, date of current version June 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3090109

Deep Speaker Recognition: Process, Progress, and Challenges

ABU QUWSAR OHI[1], M. F. MRIDHA[1], (Senior Member, IEEE), MD. ABDUL HAMID[2], AND MUHAMMAD MOSTAFA MONOWAR[2], (Member, IEEE)

[1]Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh
[2]Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

Corresponding author: Abu Quwsar Ohi (quwsarohi@gmail.com)

ABSTRACT Speaker recognition is related to human biometrics dealing with the identification of speakers from their speech. Speaker recognition is an active research area and being widely investigated using artificially intelligent mechanisms. Though speaker recognition systems were previously constructed using handcrafted statistical means of machine learning, currently it is being shifted to state-of-the-art deep learning strategies. Further, deep learning being a fast-paced domain, an absence of comprehensive survey is observed in the current deep speaker recognition technologies. In this paper, we focus on deep speaker recognition technologies. The paper particularly introduces a taxonomy, explains the progress, architectural strategies and processes of some distinctive approaches. Further, the manuscript classifies and enlists the currently available datasets and programming tools. Finally, the paper investigates the challenges and future directives of deep speaker recognition technology.

INDEX TERMS Speech processing, speaker recognition, deep learning, end-to-end architectures, meta learning.

I. INTRODUCTION

Deep learning (DL) deals with determining optimal parameters of linear and non-linear functions using gradient descending and optimization techniques. Generally, DL techniques require training on datasets, resulting in the overall strategy to be a supervised learning process. However, researchers have revealed new means of modifying the overall prediction process to be unsupervised [1]. DL techniques earned their resurgence due to superior accuracy and automated feature engineering. Yet, the current limitation of DL strategies involves unexplainable DL models along with theoretical backlog. Despite such limitations, industries holding large datasets are confidently depending on DL methods due to excellent prediction capabilities [2].

Researchers further leverage the robustness of DL frameworks, which is also perceived in the speaker recognition domain. Often implementing speaker recognition process with DL strategies are recognized as deep speaker recognition (DSR) [6]. DL is still exploring its capabilities in various domains, which also includes speaker recognition. As a result, a fast-paced improvement is observed, and new ideas are anticipated. However, DSR being the new platform of speaker recognition domain, previous strategies will obsolete shortly. Further, the theoretical grounding of ML and DL based speaker recognition is rather divergent. Therefore, DL-based surveys are required to illustrate the DL-based speaker recognition system's current scenario properly.

Speaker recognition systems started gaining success through the implementation of Gaussian mixture models (GMM), Hidden Markov models (HMM), and universal background models (UBM). The mixture of the GMM-UBM system leads to the invention of *i-vector*, which is still being used as a speaker recognition baseline in various platforms. However, deep learning methods have proved to be the current state of the art in computer vision [7], language processing [8], communication [9], networking [10], fault-detection [11], and many other platforms. As a result, researchers are currently focused on speaker recognition systems using deep learning.

Although the interest in speaker recognition systems is being shifted towards DL, surveys related to previous machine learning systems are becoming antiquated. DL architectures require new theories, ideas, and architectural skills for building robust DL frameworks. Nevertheless, limited effort has been given exploring the architectural

The associate editor coordinating the review of this manuscript and approving it for publication was Zhen Ren.

TABLE 1. The table illustrates a comparison among some of the latest surveys targeting deep learning based speaker recognition strategies.

Survey		Taxonomy	Datasets	Tools	Challenges	Deep learning networks			
Ref.	Year					Meta-learning	Residual	Adversarial	Custom
[3]	2019	✗	✗	✗	✗	✗	✓	✓	✓
[4]	2020	✗	✓	✓	✗	✗	✗	✓	✓
[5]	2021	✗	✓	✗	✗	✗	✓	✓	✓
Ours	-	✓	✓	✓	✓	✓	✓	✓	✓

perspectives of DL frameworks, which would support researchers in the current DL platform (compared in Table 1).

This paper extensively focuses on deep speaker recognition strategies. Moreover, the paper illustrates the general training methodologies, neural architectural analysis, the current drawbacks of DSR, along with future directives. The paper focuses on conceptually analysing well-acknowledged DSR research work while ignoring statistical analysis and precision comparisons of the research domain. The research highlights of the paper is as follows:

- The paper significantly focuses on presenting deep learning based speaker recognition procedures and presents a taxonomy.
- The paper elaborates on some of the notable deep learning approaches, explains the workflow and architectural concepts of networks, including multi-stage networks, end-to-end networks, generative networks [12], and meta-learning [13] in speaker recognition.
- The paper explores programming tools and datasets available for speaker recognition. The datasets are also classified on distinct speaker recognition research domains.
- The paper further explores the challenges of current speaker recognition systems, outlines an ideal concept and examines the opportunities.

In speaker recognition domain, surveys have been conducted targeting feature extraction methodologies [14], [15], speaker recognition stages [16], domain adaptation of speaker recognition [17] etc. The survey papers mostly investigate outdated machine learning methodologies. Currently, there exists minimal investigation in DL-based speaker recognition methodologies. No paper completely provides a compact survey of the DL-based speaker recognition system to the best of our knowledge. Table 1 compares the latest surveys conducted on a similar domain, which is compared based on different aspects, including datasets, programming tools, taxonomy, critical analysis, and deep learning sectors. Apparently, the survey covers every dimension of survey methodology.

This paper is specifically targeted at researchers and engineers interested in the perspective of DSR. Further, while describing such strategies, it is assumed that readers bear the basic knowledge of speech signal processing and deep learning fundamentals.

The rest of the sections are organized as follows. Section II provides motivation on speaker recognition systems by introducing various applications. Section III introduces the fundamentals of speaker recognition and the architectural taxonomy of deep speaker recognition systems. Section IV explains the stage-wise speaker recognition systems. Section V explains the end-to-end strategies based on DL architectures. Section VI notes out the available programming tools and datasets available for speaker recognition implementation and evaluation. Section VII points out a novel speaker recognition system that points out the challenges with suggestions. Finally Section VIII concludes the paper.

II. APPLICATIONS OF SPEAKER RECOGNITION SYSTEMS

Speaker recognition has numerous applications in various fields related to authentication, re-identification, which directly depends on human biometric features. Also, speaker diarization has a broad usage domain. Robust systems can be developed using speaker diarization while it is merged with speech recognition systems. In this section, several applications of speaker recognition systems are discussed.

Speaker recognition has a greater impact on telecommunication systems [18]. Speaker recognition can identify unknown callers based on their existing voice profiles in a particular database. Also, the recognition system can be implemented to automatically block unknown callers using an existing database of known voice profiles.

In the banking sector, speaker recognition systems can be used for authentication. Users can be easily verified over the phone and dealt with appropriately in telemetric banking transactions. Call centers can also inherit speaker re-identification for personalized services and queries. In the case of personalization, speaker recognition systems have a significant impact on IoT (Internet of Things) devices.

Digital assistants and smartphones often require a verification system for users. Using speaker verification, digital assistants can easily verify the actual speaker. In contrast, smartphones can also inherit speaker verification systems to unlock without even touching and facing the camera. In the case of a multi-user platform, a device can recognize multiple users and provide personalized features to each individual through speaker identification.

Search engines now inherit speech recognition systems to search on the web through listening through the microphone.

For personalized search features, search engines often save various information. However, search engines only identify users only when they use specific accounts. If speaker recognition systems are implemented on search engines, personalized search results can be directly provided to the user by identifying their voice. In such a case, no account information or password-based authentication is required for a personalized search.

Speaker diarization with speech recognition can provide person-specific transcripts. Such systems impact law courts, lawyers, judges, or even generating automated transcripts of corporate meetings. Often such systems help to develop automated apps for taking notes on any particular conversation. Also, it is possible to develop applications that may trigger a panic alarm based on any critical condition through hearing speech and identifying the speakers.

The speech data can serve as a human biometric similar to a fingerprint or facial image at a national level. Hence, fraud calls can be tracked via a speaker recognition system. Also, speaker recognition systems can be used for forensic purposes. Student attendance system, employee attendance systems can be developed using speaker recognition methods.

As speech is a human biometric, speaker recognition systems can be implemented in numerous applications requiring identification, verification, and justification of individuals using voice. As speech is the most appreciatable communication method, speaker recognition systems significantly impact technologies dealing with voice and voice-based communication. Hence, speaker recognition systems have incredible applications in human biometrics and require extensive investigation towards generating robust speaker recognition systems.

III. SPEAKER RECOGNITION FUNDAMENTALS

The general training process of deep learning architectures includes two phases, training and testing. In the case of speaker recognition methods, the training phase is often defined as enrollment. The speaker recognition platform has such distinguished mechanisms that are described in this section. Moreover, to properly construct such mechanisms' instinct and facilitate the readers, the mathematical notations used in the paper are illustrated in Table 2.

A. TYPES OF SPEAKER RECOGNITION METHODS

Based on the recognition policy and environmental state, speaker recognition processes can be classified into the following sub-domains, based on the recognition criteria:

- **Speaker verification:** Speaker verification deals with authenticating users based on their voices. Intuitively, a deep speaker verification mechanism is a binary classification problem. A speaker verification model learns a specific set of weights w_f such that a DL function f is optimized so that minimal value of the below-mentioned equation can be achieved:

$$\mathcal{L}(f_{w_f}(x), y) \tag{1}$$

TABLE 2. The mathematical notations used in the paper are summarized.

Notation	Description
$\mathcal{L}(\cdot,\cdot)$	The loss function used to train a DL architecture. The former parameter receives the DL models output, while the latter receives the ground/actual output.
$(\cdot)^e$	Any specific data used in enrollment or training time.
$(\cdot)^t$	Any specific data used in testing time.
$(\cdot)^T$	Transpose of a matrix.
N	Number of data in the data-batch.
C	Number of class (unique speakers) in the data-batch.
$\mathcal{D}$	Enrollment dataset, $(X_i^e, y_i^e) \in \mathcal{D}$.
f	A DL network.
W_f	Weights for DL network f.
$\mathbb{R}^{n\times m}$	Real number of dimension $n \times m$.
$\mathbb{E}_x$	Expected value of appearing x from a random instance.
$*$	Convolution operator.
$\circ$	A element-wise multiplication (Hadamard) operator.

Here $f_{w_f}(\cdot)$ defines a general DNN model containing a set of weights w_f and outputs predictions $\hat{y}$ for which the loss generated by the function $\mathcal{L}(\cdot,\cdot)$ is minimal.

- **Speaker identification:** Speaker identification is a multi-class classification problem. The task is to identify speakers for a given utterance where they can be anyone from a registered set of speakers. However, if the objective is to find only one specific speaker, it becomes similar to a speaker verification method. Further, constructing an ensemble of speaker verification methods can result in generating a speaker identification mechanism. Mathematically the identification method can be derived as,

$$\mathcal{L}(\underset{y}{\text{argmax}}\{f^1_{w_{f1}}(x), f^2_{w_{f2}}(x), \ldots, f^n_{w_{fn}}(x)\}, y) \tag{2}$$

Here $f^i_{w_{fi}}(\cdot)$ denotes the DNN classifier recognizing the probability of occuring i^{th} speaker, where $i \leq n$. As, it is observed that a speaker identification being a superset of verification methods, identification and verification systems are utilised interchangeably.

- **Speaker diarization:** Speaker diarization is a similar instance of a speaker identification system. Speaker diarization is mostly required for automated speech transcription systems, where dialogue is generated along with speakers' information. As multiple speakers can speak at once, speaker diarization is considered as a multi-output/multi-label classification problem. Moreover, speaker diarization systems do not have any enrollment process and identifies speakers unsupervisely.
- **Speaker recognition in-the-wild:** 'In-the-wild' [19], [20] scenario refers to a real-world scenario of

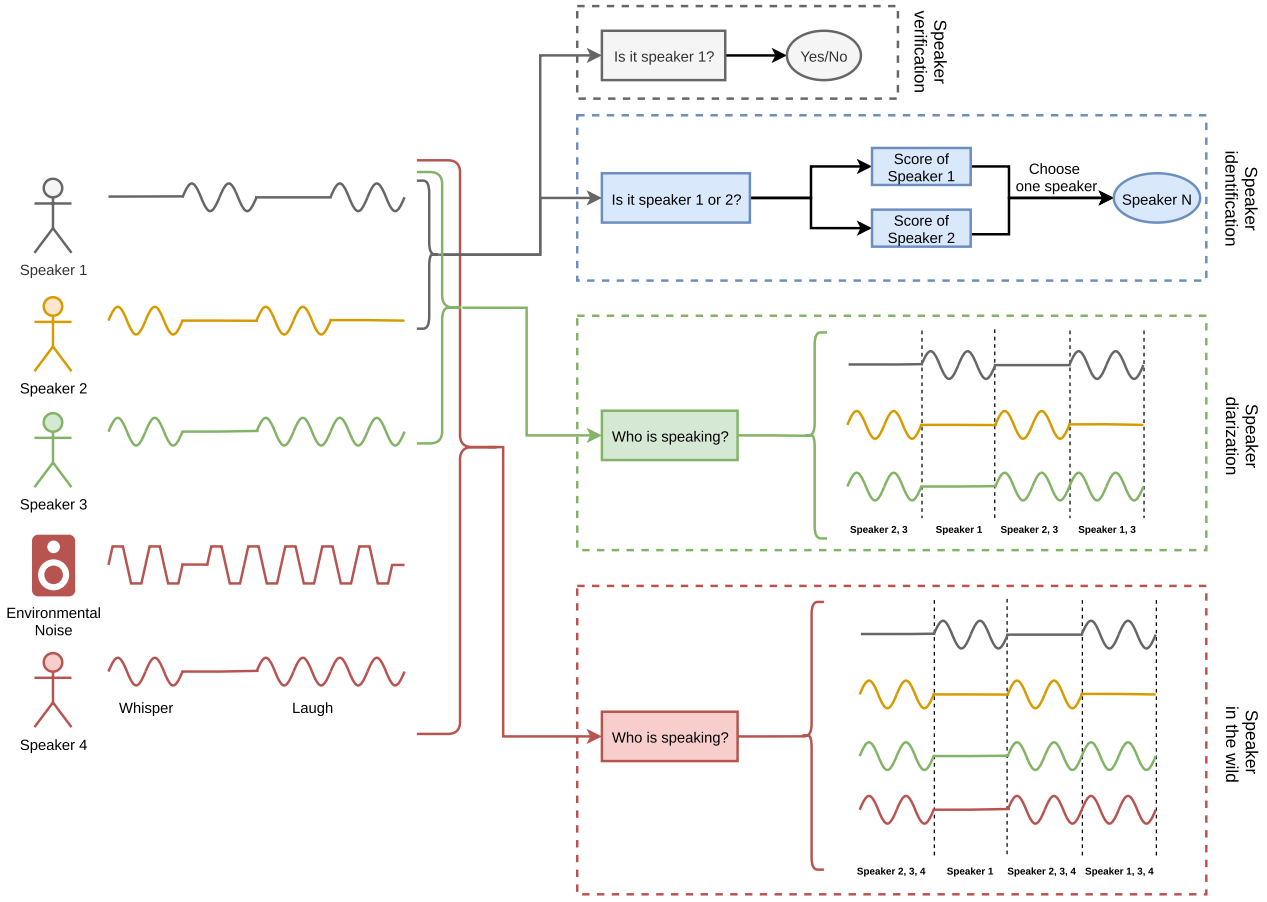

FIGURE 1. The figure illustrates the fundamental difference of speaker verification, identification, diarization, and 'in-the-wild' recognition strategies.

speaker recognition systems in which conditions are unknown. 'In-the-wild' scenario may contain data (input speech) similar to speaker verification, identification, and diarization. Further, 'in-the-wild', speaker recognition datasets contain numerous corrupted conditions: noise, laughter, channel effects, music, echo, cross-talks etc. 'In-the-wild' challenge is currently being explicitly targeted by the researchers.

Figure 1 visualizes the common differences of speaker recognition policies. The general types of speaker recognition systems can be divided into two categories based on the recognition rule: (a) text-dependent and (b) text-independent. Text-dependent systems can only recognize speakers if they speak specific phrases. Oppositely, text-independent systems are independent of such phrase limitations. Text-dependent systems are becoming obsolete as users often consider the text-dependency as a limitation.

B. ARCHITECTURAL PERSPECTIVE

Deep learning based speaker recognition methods can process inputs in two particular input patterns: a) directly process raw sound waves, b) pre-processed data. In processing raw sound waves, speech frames may be normalized based on the actual sound wave or thresholds. Architectures like SincNet, RawNet, AM-MobileNet (described in Section V-B1, V-A2, V-A3 respectively) is trained directly using raw speech data. However, most systems depend on pre-processing to transfer time domain data to time vs frequency domain data.

As speech data is represented in a signal of continuous time, they are generally sliced into shorter segments while training. In speaker recognition literature, utterance refers to a window of sounds containing a single word pronouncement. In contrast, a frame refers to a smaller portion of speech data (20-500 milliseconds) that can include a complete utterance or a part of an utterance. In general, speech frames are used as an input of the recognizer (with/without pre-processing), containing phoneme information. Enrollment and testing are conducted by giving speech frames of equal length to the recognizer. The pre-processing of speech data is observed to be of three types: a) spectrogram, b) mel-filterbank, and c) MFCC. Figure 3 illustrates the workflow for extracting the features.

DL is extensively applied in speaker recognition systems. Researchers are discovering numerous aspects of speaker recognition mechanisms due to the advancement of datasets and DL strategies. Hence, numerous techniques have been

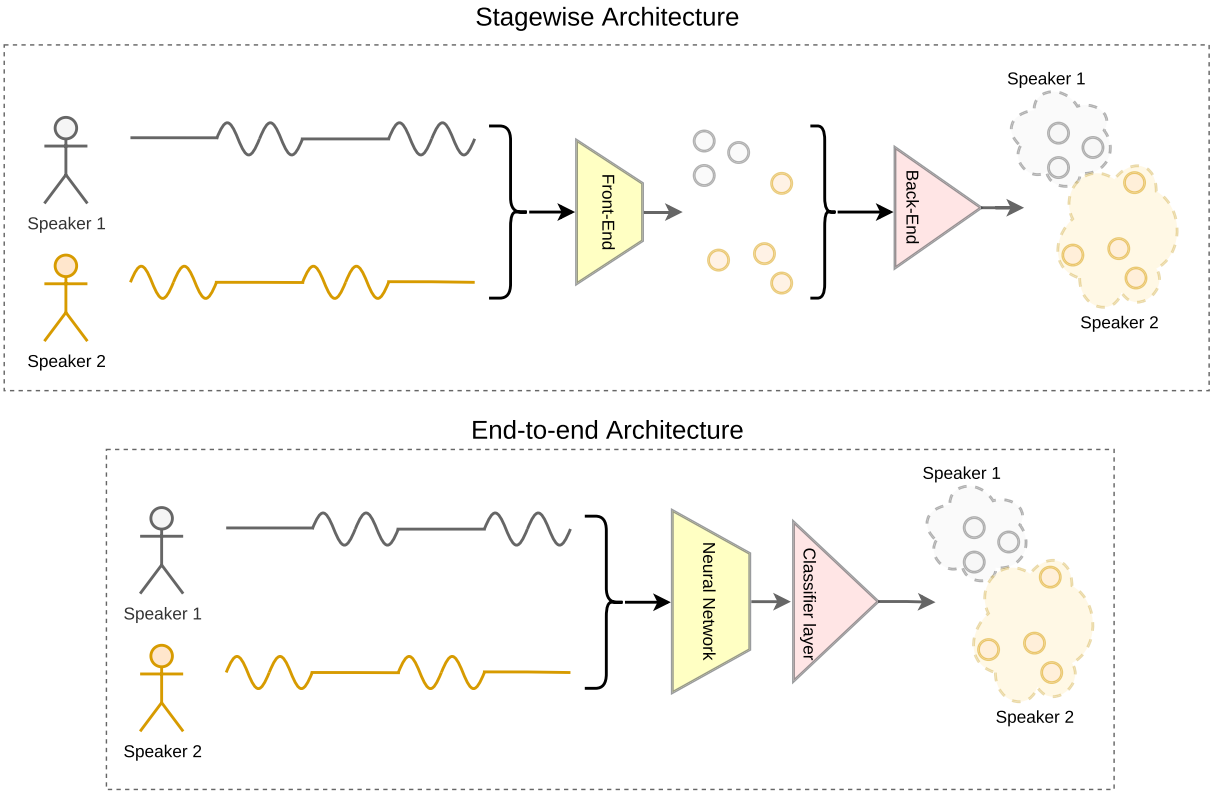

FIGURE 2. The figure illustrates the difference between stagewise and end-to-end architecture. In stagewise architecture, the front-end converts speech to low-level embeddings, further classified using back-end models. The front and back ends are trained separately. In contrast, end-to-end architecture is a combined pipeline and also trained jointly.

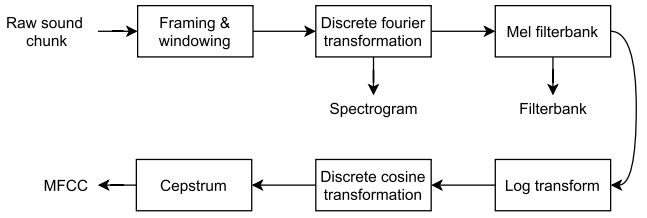

FIGURE 3. The figure illustrates the steps required to extract spectrogram, filterbank, and MFCC from a raw audio wave. Such pre-processing techniques do not require any trainable weights, hence can be easily fused with neural networks.

explored in speaker recognition systems. This paper examines DL methods' architectural strategies covering the overall speaker recognition domain as much as possible. Figure 4 illustrates a taxonomy of the deep learning methods investigated in this paper.

In the taxonomy, the speaker recognition domain is classified into stage-wise and end-to-end strategies. The stage-wise strategy involves two stages: speaker-specific feature extraction and classification of speakers. The first is defined as the front-end, and the latter is defined as the back-end of a DL network. Section IV extensively investigates such strategies. All of the speaker embedding systems which maps the dissimilarities of speech features are presented in stage-wise architecture.

End-to-end systems do not require a multi-stage network; however, they may require additional pre-training for better results. Discriminative models (residual and custom layered networks) can be jointly trained to produce better results. In contrast, generative networks require pre-training for better initialization. Finally, these methods are jointly trained. The core difference between stage-wise and end-to-end systems points to the training strategy. In most stage-wise techniques, the front and back ends are separately trained. In contrast, end-to-end systems are jointly trained at a specific phase. Section V thoroughly explains the end-to-end deep speaker recognition systems. Figure 2 illustrates the common difference between end-to-end and stagewise architecture.

End-to-end systems are segmented into four categories: residual networks, networks with custom layers, generative networks, networks involving meta-learning strategies. Residual networks are commonly observed in image processing [21], which is greatly used in other domains for its effectiveness against vanishing gradient problem. In speaker recognition systems, some networks contain customized layers to capture speaker features, which is pointed to as a different branch of end-to-end speaker recognition in this paper (described in Section V). Generative networks include a variety of usefulness, hence defined as a separate instance of end-to-end strategy (described in Section V-C). Finally, meta-learning involves a different approach of learning from

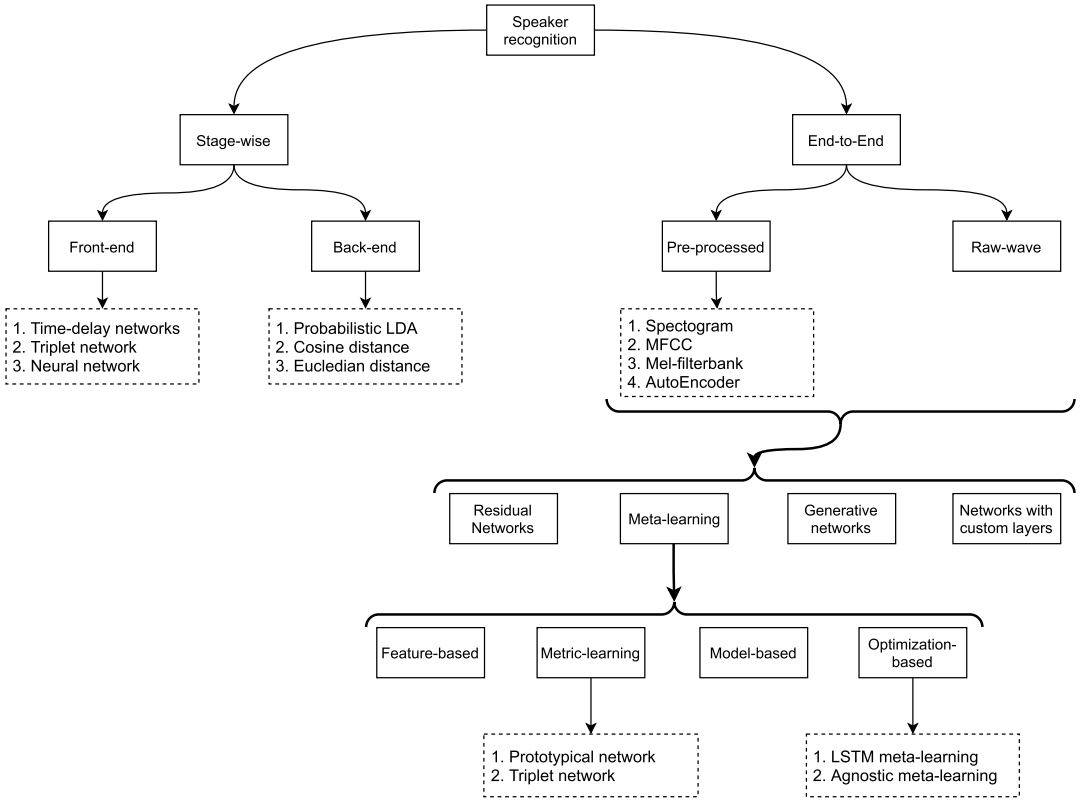

FIGURE 4. The figure illustrates the taxonomy observed in deep speaker learning, investigated in this paper.

less data. Meta-learning further contains sub-branches, which are briefed in Section V-D. The following sections describe the architectures illustrated in the taxonomy.

IV. STAGE-WISE DEEP SPEAKER RECOGNITION

DL architectures substantially auto-extract features from input data, whereas machine learning technologies required handcrafted feature extraction processes. Before the extensive DL usage, speaker recognition procedures required manual extraction of speech-related features until the *i-vector* appeared [22]. *i-vector* is an ML approach containing a feature extractor or frontend implemented using GMM-UBM, with a probabilistic linear discriminant analysis (PLDA) [23] used as a classifier or backend. In the machine learning approach, a feature extractor refers to a function that extracts prominent values suitable for discriminating among multiple output classes. The early success of *i-vector* based systems lead to many hybrid methods combining *i-vector* with DL architectures [24], [25]. However, such methods are rarely observed in the current scenario, and being out of the scope of DL architectures, we exclude briefing *i-vector* mechanisms.

Being inspired by *i-vector*, researchers implemented DL based speaker embedding systems, *d-vectors* [26], *x-vectors* [27], and *t-vectors* [28]. Such embedding systems are mostly used to construct stage-wise architectures [29]. In the following Sections IV-A to IV-C, such deep speaker embedding systems are described. Section IV-D describes the back-end srategies.

A. DEEP VECTOR: D-VECTOR

Deep vectors or *d-vectors* are trained using frame-level speech information. The *d-vector* architecture inputs 300*ms* speech frames, containing 40 filterbanks. Further, the network contains four dense (or fully connected) layers with 256 nodes per layer. The last two hidden layers contained *dropout* of 0.5, which works strongly with *maxout* DNN [30]. *dropout* layers give better generalization while training DL architectures on small datasets. The initial *d-vector* architecture contained approximately 0.6 million parameters.

Initially, the *d-vector* is trained as a classifier by connecting a softmax activation layer. Figure 5 illustrates the architecture of *d-vector*. As the architecture contains softmax activation layer, the enrollment process is conducted using softmax loss [31] function stated as:

$$\mathcal{L}_s = -\frac{1}{N}\sum_{i}^{N} log \frac{e^{W_{y_i}^T f(x_i)+b_{y_i}}}{\sum_j^C e^{W_j^T f(x_i)+b_j}} \tag{3}$$

In the equation, N is the batch size of enrollment, C is the number of class (unique persons) in the enrollment dataset, $f(\cdot)$ is the embedding function's produced output. The W and b are the weight and bias of the softmax layer. After completing initial training, the softmax layer is left

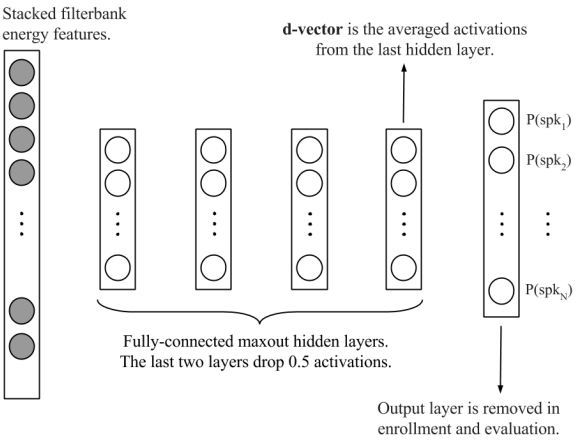

FIGURE 5. The leftmost layer receives speech filterbank energies and passes through the four densely connected layers of 256 nodes. The rightmost (output) layer is used as a classifier layer, attached with a softmax activation function. After enrollment, the outer layer is excluded.

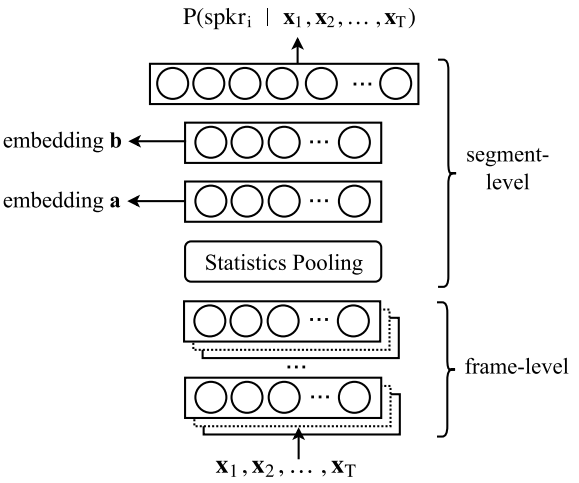

FIGURE 6. The network receives 24 filterbanks data produced from 25*ms* audio data. The frame-level extractors are five time-delay layers, followed by a statistical pooling layer. Embedding a and b are dense layers both produces speech embeddings. The output layer is a dense layer used for softmax layer.

out. *d-vector* is produced by averaging the activation of the last hidden layer of the architecture. The *d-vector* system is trained using frame-level information, generating frame-level embeddings. Further, the utterance-level features can also be extracted by the having mean of a particular utterance's frame-level features. Finally, PLDA is added that receives *d-vector* systems output and classifies speakers.

B. TIME-DELAY APPROACH: X-VECTOR

x-vector [27] inputs variable-length speech input and produces speaker embeddings. The *x-vector* architecture is implemented using a stack of time-delay [32] layers, further passed to statistical pooling function. The first time-delay nodes of the network receives 25*ms* speech frames (24 dimentional filters). The statistical pooling produces segment-level (or utterence-level) features by calculating the mean and standard deviation of the previous time-delay layer. The statistical pooling function is followed by two fully connected layers, considered as embedding *a* and *b*, respectively. Similar to *d-vector*, the *x-vector* system is also trained by attaching a fully connected layer with a softmax layer. Data augmentation is executed in the training phase of *x-vector* to boost the generalization capability. Excluding the last softmax layer, the architecture contains 4.2 million parameters. Figure 7 explicates the *x-vector* system's network architecture. The *x-vector* systems are trained using softmax loss function as explained in equation 3 [28]. *x-vector* successfully beats *i-vector* systems and *d-vector* systems by achieving lower equal error rate (EER).

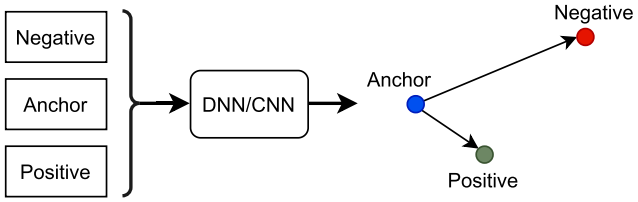

FIGURE 7. Triplet architecture contains a DNN/CNN network (shared into three networks for three parallel inputs) that receives three inputs. The negative input is dissimilar to the anchor, while the positive input belongs to the same speaker as the anchor data. The network learns to keep anchor and positive data's embedding closer, keeping the negative data's embedding farther than a constant margin of α.

C. TRIPLET NETWORK: T-vecTor

After the success of FaceNet [33] embeddings specifically implemented for facial images, researchers have implemented such a method in various domains. Such architecture is implemented based on a triplet loss architecture trained based on three inter-relatable inputs described in Figure 7. Triplet architectures are trained using a shared DNN triplet network and usually trained using triplet loss defined as,

$$\mathcal{L}_T = \sum_{i}^{N}(||f(x_i^a - f(x_i^p))|| - ||f(x_i^a) - f(x_i^n)|| + \alpha) \quad (4)$$

Here x_i^a, x_i^p, x_i^n defines the anchor, positive, and negative data from the enrollment dataset. The function $f(\cdot)$ produces the embeddings for a given speech frame. α is a margin that is enforced between a positive (anchor-positive) and negative (anchor-negative) pairs. Learning to disciminate against positive and negative pairs, the triplet-network learns to cluster

After enrollment, the network receives a single speech input and produces speaker embeddings, represented as *t-vector* [28]. In comparison to *x-vector* systems, *t-vector* does not perform superior yet closely compete against *x-vector* systems [17]. The embedding dimension of *t-vector* is similar to the last dense/fully-connected/linear layer's output dimension. It is observed that the last layer does not attain any non-linear activation functions. The architectural constrains of the *t-vector* network is similar to a siamese network [34].

TABLE 3. The table illustrates a systematic comparison of different embedding systems.

Embedding	Front-end		Back-end	
	Training strategy	Loss	Back-end name	Training strategy
i-vector	Unsupervised	-	PLDA	Supervised
d-vector	Supervised	softmax loss	cosine	Unsupervised
x-vector	Supervised	softmax loss	PLDA	Supervised
t-vector	Supervised	triplet loss	PLDA,cosine, eucledian-distance	Supervised/Unsupervised

D. ANALYSIS OF NETWORK BACK-ENDS

Table 3 illustrates the back-end strategies adopted for the front-end architectures. Implementing PLDA as a back-end requires separately training the PLDA based on the speaker vector outputs. In contrast, the cosine (derived in equation 5) and euclidean-distance back-ends do not require any training because both back-ends produce final output base on the distance metric.

Amongst the various DL based embedding systems, *x-vector* performs superior in most cases [17]. *t-vector* closely competes against *x-vector* architecture. *i-vector* and *d-vector* architectures becoming obsolete as they lack behind in comparison of performance. However, *t-vector* architectures have further possibilities if proper architectural investigation is conducted. Further, *t-vector* architectures may result superior if proper augmentation policies are adopted compared to *x-vector*.

The challenge of such speaker embedding systems is the limitation of proper domain adaptation between enrollment and test dataset. Such embedding systems may suffer from erroneous embeddings due to the scarce of adequate data. However, augmentation strategies help to reduce the variance of the domain between enrollment and test dataset. Consequently, efforts have been made to reduce the interpretation of unseen data by improving training policies [35], introducing generative strategies [36], and so on. Nevertheless, such speaker embedding systems are becoming outdated, specifically for the supervised recognition task. DL architectures' vast evolution has led to end-to-end systems that do not require stage-wise training. A single DL architecture efficiently handles all of the recognition tasks.

V. END-TO-END DEEP SPEAKER RECOGNITION

An end-to-end architecture learns directly from the input without the necessity of front and back-end strategies. End-to-end architectures are trained with a single loss function, and the whole architecture is jointly trained. In most cases, end-to-end systems are trained only once. In contradiction, in the deep speaker embedding approach, the front-end and back-end systems are trained separately. However, the end-to-end architectures require an extensive amount of training data to learn the feature representations [38]. Moreover, designing such end-to-end architecture requires DL skillsets and often requires customized equations to learn from input data properly. Hence, researchers need to focus on DL strategies and problem interpretation intensely. The end-to-end strategies are classified into four different regions (illustrated in Figure 4), elaborated in the below sections.

A. RESIDUAL NETWORKS IN DEEP SPEAKER RECOGNITION

Residual networks are widely implemented in speaker recognition tasks and introduced in both speaker feature extraction [37], [39] and end-to-end systems [40]. Residual networks actively eliminate the vanishing gradient issue and solve the reduction of accuracy while increasing layers in DL architectures.

1) DeepSpeaker: END-TO-END SPEAKER EMBEDDING

DeepSpeaker [37] architecture is the early implementation of end-to-end architecture. Although the architecture was specially designed for speaker embeddings, it has pioneered end-to-end speaker recognition systems [41]. The DeepSpeaker contains basic CNN and residual network, proved deep architectures are enough to recognize speakers. Figure 8 explains the flow of the model. Apart from CNN and residual networks, the architecture performs a temporal average (averaging over the time dimension) to convert frame level to utterance level information. It further transforms utterance level information into 512-dimensional feature vectors, which is normalized to 1. The architecture is trained using triplet loss (defined in equation 4). However, general triplet loss architectures implement euclidian distance, whereas DeepSpeaker implements cosine distance between pairs stated as:

$$COS(x_i, x_j) = \frac{x_i^T x_j}{||x_i||^2 ||x_j||^2} \tag{5}$$

Here x_i and x_j is the 512-dimensional speaker embeddings. As the architecture performs a unit norm operation it is guranteed that $||x_i||^2 = ||x_j||^2 = 1$. Although the performance of the architecture lags compared to the current implementations, the method revealed that speaker recognition performance is not sensitive to stride in the time dimension.

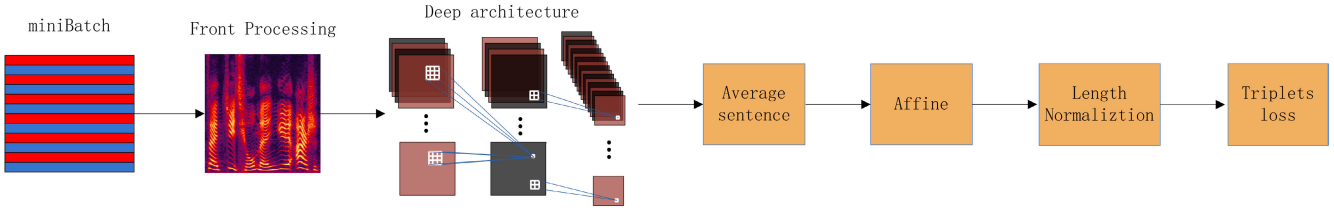

FIGURE 8. The figure illustrates the process of DeepSpeaker [37] architecture. The method receives 64-dimensional 0.25 ms filterbanks. The filterbanks are further processed using CNN and residual blocks. Later, the outputs are averaged in the time domain, passed to a 512-dimensional dense layer and unit normalized. The overall method is trained using triplet loss.

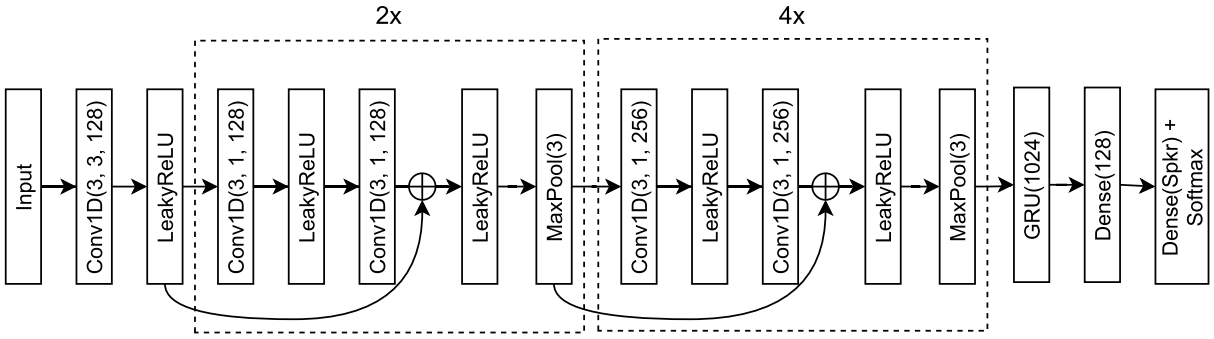

FIGURE 9. The figure represents RawNet's architectural diagram. The naming parameter conventions used for layers are: ***Conv*(*kernel_size*, *strides*, *channels*)**, ***Dense*(*nodes*)**, ***GRU*(*nodes*)**, ***MaxPool*(*pool_size*)**. The architecture contains 2 and 4 repetitive residual blocks confined in dashed rectangles.

2) RawNet: RESIDUALS AND RECURRENT NETWORK

RawNet architecture [42] is implemented based on a convolutional architecture with residuals. Apart from the SincNet architecture (described in Section V-B1), RawNet does not include any customized layers, while recieving raw-waveforms like SincNet. Instead, RawNet architecture proves speaker recognition can be decently handled using pre-existing methodologies. Figure 9 illustrates the RawNet architecture. The model includes residual blocks for speaker identity mapping [43] and facilitate the training process. The RawNet architecture is firstly trained by adding a global average layer, removed after the training process, to reduce overfitting. From an architectural perspective, LeakyReLU with less residual blocks reduces overfitting chances while boosting accuracy. Further, instead of the previous general usage of LSTM layers [44], a GRU layer is attached to find the inter-relativity of time-based features. The RawNet architecture requires 5.7 million parameters in total.

The RawNet architecture is trained using an accumulated loss function described as:

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_C + \mathcal{L}_{BS} \tag{6}$$

$$\mathcal{L}_{CE} = -\sum_{i}^{C} y_i \, log \, f_w(x_i) \tag{7}$$

$$\mathcal{L}_C = \frac{\lambda}{2} \sum_{i}^{N} ||x_i - c_{y_i}||$$

$$where, \quad \lambda = 10^{-3} \tag{8}$$

$$\Delta c_{y_i} = \frac{\sum_{i=1}^{N} \delta(y_i = i)(c_{y_i} - e_i)}{1 + \sum_{i}^{N} \delta(y_i = i)}$$

$$where, \quad \delta(condition) = \begin{cases} 1, & if \ True \\ 0, & otherwise \end{cases} \tag{9}$$

$$\mathcal{L}_{BS} = \sum_{i}^{N} \sum_{j, j \neq i}^{N} cos(W_i, W_j) \tag{10}$$

The $\mathcal{L}_{CE}$ is cross-entropy loss derived in equation 7. $\mathcal{L}_C$ and $\mathcal{L}_{BS}$ is the centre loss [45] and basis loss [46], sequentially. The centre loss reduces the intra-class covariance while the basis loss function reduces the inter-class covariance. c_{y_i} is the centre calculated for the centre loss, updated by a rate α multiplied a delta centre value as derived in equation 9. The W_i is the basis embedding for the $i'th$ class derived in euqation 10.

3) AM-MobileNet: MobileNet IN SPEAKER RECOGNITION

MobileNet architectures [47], [48] are specifically designed for mobile devices, requiring lower memory and computation cost. MobileNet architectures strongly rely on depthwise separable convolution that reduces the computational and memory complexities. Depthwise separable convolution performs a depthwise convolution followed by a pointwise convolution. MobileNetV2 [48] further strengthens the structure of the architecture by fusing residual networks. Instead of depending on general residual blocks [21], MobileNetV2 inverted residual blocks that significantly

reduces the number of parameters. Further, instead of traditional ReLU activation, MobileNet architectures implement ReLU6 activation function derived as:

$$ReLU6(x) = min(max(0, x), 6) \tag{11}$$

The ReLU6 function limits the activation values within [0, 6], which require 3 bits for the left decimal points, guaranteeing precision for the right decimal values.

AM-MobileNet1D [49] adapts the strategy of MobileNetV2 architecture, by replacing 1D convolutions instead of 2D convolutions. Hence, the resulting architecture is suitable for recognizing one-dimensional inputs. The model contains 2.8 million parameters in total.

B. NETWORKS WITH SPEECH SPECIFIC LAYERS

Some architectures implement a layer or a set of layers (as a network), providing specific advantages to DL architectures to speaker recognition architectures. This segment of speaker recognition models covers architecture like SincNet (discussed in Section V-B1) are directly based on sound processing. At the same time, architectures like autoencoder (discussed in Section V-B3), VLAD (discussed in Section V-B2) are adapted from different domain of DL.

1) SincNet: PARAMETERIZED SINC FUNCTION

SincNet [50] architecture depends on bandpass filters for waveform feature extraction. Bandpass filters only recognize certain waves frequencies. SincNet architecture extracts vital audio features using such bandpass filters, which are dependent on a specific person. SincNet architecture implements sinc functions ($sinc(x) = \frac{sin(x)}{x}$) to perform convolutions in the initial layers, that can mathematically state as:

$$\begin{aligned} y[n] &= x[n] \times g[n, a_1, a_2] \\ g[n, a_1, a_2] &= 2a_2 \frac{sin(2\pi a_2 n)}{2\pi a_2 n} - 2a_1 \frac{sin(2\pi a_1 n)}{2\pi a_1 n} \end{aligned} \tag{12}$$

Here, $x[n]$ is the input signal and $y[n]$ is the filtered output. g is a bandpass filter, where a_1 and a_2 is the low and high cutoff frequencies. After applying bandpass convolution, the filtered output is passed to a general DL architecture to perform the classification task. Figure 10 illustrates the overall SincNet architecture. The advantage of such filters is it requires a low number of parameters, compared to 1D-CNN. For F number of filters, each signal being a length of L, a 1D-CNN requires $F \times L$ parameters. In contrast, sinc layers require only $2F$ parameters. Further, SincNet architecture achieves better performance and faster convergence compared to CNN/DNN based method with or without additional pre-processings. The initial SincNet architecture was implemented using softmax loss (Equation 3) function. However, further investigation [51] points out softmax loss function obstructs SincNet architecture's robustness. Therefore, the usage of AM-softmax loss is implied. AM-softmax loss can be represented as:

$$\mathcal{L}_{AMS} = -\frac{1}{N} \sum_{i}^{N} \frac{e^{s(W_{y_i}^T f(x_i) - m)}}{e^{s(W_{y_i}^T f(x_i) - m)} + \sum_{j, j \neq y_i}^{C} e^{W_j^T f(x_i)}} \tag{13}$$

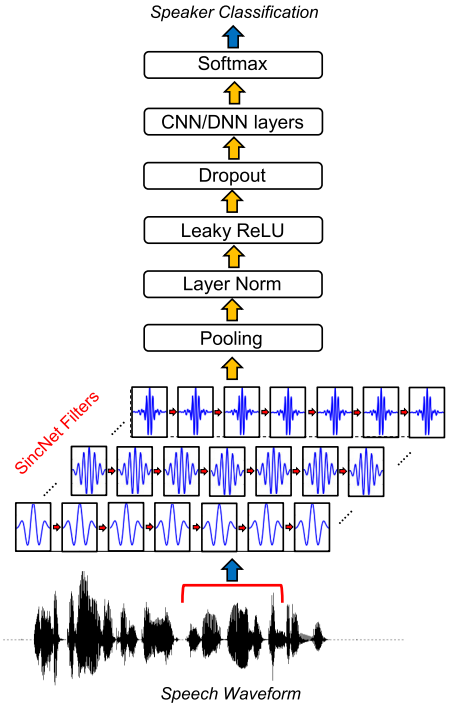

FIGURE 10. The figure depicts SincNet architecture's configuration. Speaker-dependent frequencies are extracted using bandpass filters. Pooling, normalization and Leaky ReLU is used for non-linearity. Further information is passed to DNN/CNN layers to derive high dimensional information.

Here, s is a scaling factor, learned through backpropagation. m is the margin of AM-softmax loss, which is introduced into the loss function. W is the last weight of the last layer, and bias values are ignored. $f(\cdot)$ is the embeddings produced by the DL architecture, excluding the last layer.

2) VLAD ARCHITECTURES IN SPEAKER RECOGNITION

Vector of locally aggregated descriptors (VLAD) [52] is a modification of the Fischer kernel [53], used for visual characterization and description. NetVLAD [54] is an improved version specifically used as a deep learning layer. Consequently, GhostVLAD [55] is a furnished version of NetVLAD. Both NetVLAD and GhostVLAD receive N number of D-dimensional local descriptors assigned to K clusters with cluster centers c_K. For each $j'th$ dimension of a single cluster k, the VLAD is calculated as follows:

$$V(j, k) = \sum_{i=1}^{N} a_k(x_i) \left(x_i(j) - c_k(j)\right) \tag{14}$$

Here, $a_k(x_i)$ defines the soft membership (calculated using softmax) of the embedding x_i. GhostVLAD and NetVLAD share the same formulation, except the number of clusters K is increased, which is not used in the final identification task. Xie *et al.* [41] proposed a VLAD-based utterance level aggregation strategy. The architecture includes a lightweight ResNet which's output is further computed

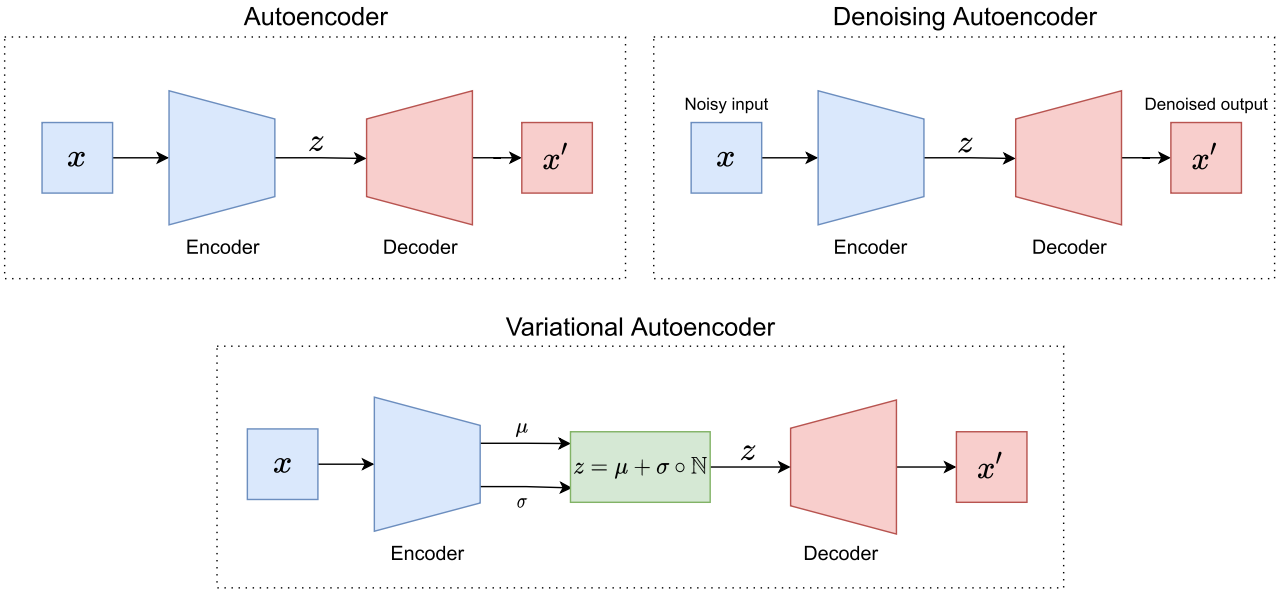

FIGURE 11. The figure illustrates three different types of autoencoders, general, denoising, and variational autoencoder, respectively. All of the architectures include an encoder and decoder. Autoencoder architectures are trained using l2-norm, $||x - x'||^2$. The variable z is the hidden low dimensional representation of input data x. Variational autoencoder contains the mean (μ) and deviation (σ) of the input data x, further represented into z.

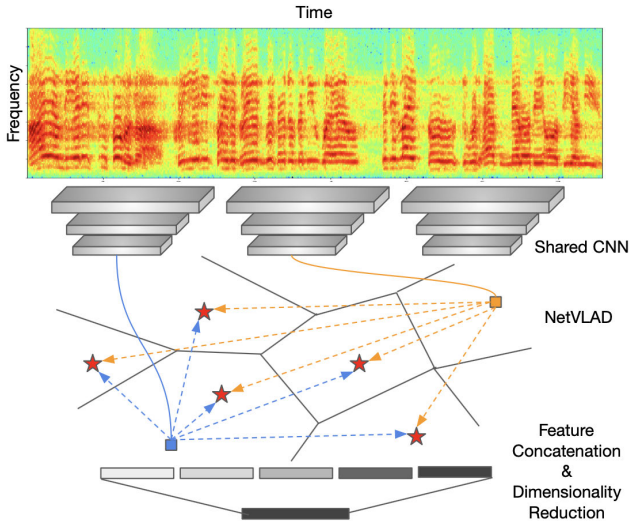

FIGURE 12. The figure illustrates the VLAD layer used to extract utterance level information proposed by Xie *et al.* [41]. A thin-ResNet extracts frame-level features, which VLAD aggregates to produce the final result.

using NetVLAD and GhostVLAD layers. The architecture receives multiple frame-level spectrograms and generates VLAD output further concatenated and reduced to generate the final output. Figure 12 illustrates the proposed architectures. The architecture produces superior accuracy using GhostVLAD. GhostVLAD includes extra cluster centers than usual (used 2 in such case), which is always ignored, generating the final result. The extra clusters recognize noises and fragments that result in avoiding erroneous recognition.

3) AUTOENCODERS IN SPEAKER RECOGNITION

Autoencoders is widely implemented in various tasks, including emotion recognition [57], age prediction [58], speech recognition [59] and so on. Amongst the various domains, autoencoders have also been introduced in the speaker recognition for data encoding, feature similarity, and denoising capability [60]. However, the implementation of autoencoders is specifically focused on speech variation reduction. Hence we investigate such implementations in this single section.

Speaker recognition systems suffer from various environmental attributes such as distance of speakers w.r.t. of the input device, noise, reverberation, etc. Autoencoders are implemented to reduce such environmental factors from speech data [60]. DAE is mainly used in speaker recognition systems to map far-field [61], and near-field speaker data to a closer representation [60]. In such a case, DAE is inputted both far and near field data extracted from speaker embedding systems. DAE learns to represent far-field and near-field embeddings to a similar representation, resulting in a marginal accuracy in complex environments.

DAE is further acknowledged as a domain balancer for speaker recognition systems [62]. An in-domain data refers to the data being similar to the testing or actual environment. Further, data that can be used for the same task but containing different environmental factors are considered out-domain data. In such a case pair of encoders are used to input in-domain and out-domain data. A single decoder learns the representation of both encoders. However, autoencoders are currently less-observed, as DL frameworks are more robust in domain adaptation.

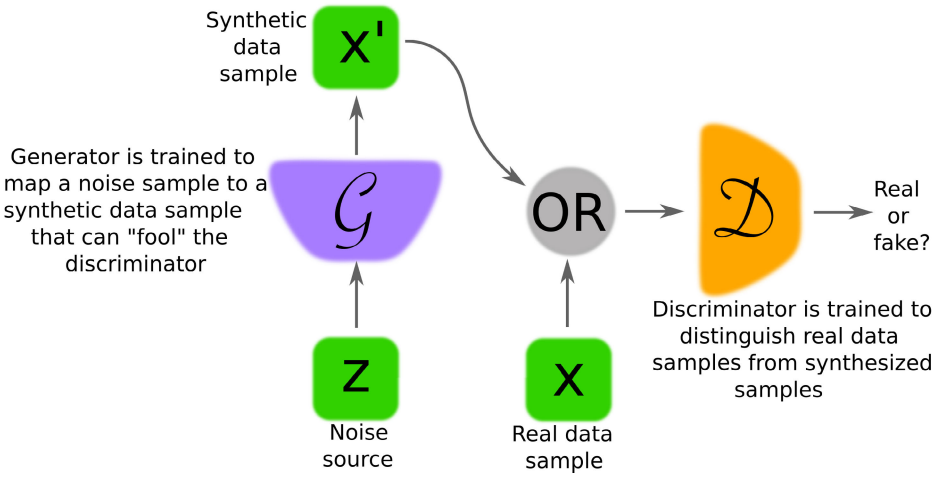

FIGURE 13. The figure illustrates the general workflow of a GAN. A generator learns to map noise samples z into a data distribution x. A discriminator tries to identify actual data and generator-outputted data. The image is adopted from the work of Creswell *et al.* [56].

C. GENERATIVE ADVERSARIAL NETWORKS IN SPEAKER RECOGNITION

In the machine learning domain, a model can be classified into two cases based on the output: a) discriminative and b) generative [63]. Discriminative models focus on classification tasks, $f(x) = \text{argmax}_y\, p(y|x)$. In contrast, generative models focus on learning the distribution of a particular dataset, $f(y) = \text{argmax}_y\, p(x|y)\, p(y)$.

Generative methods have been widely explored in the speaker recognition domain. Specifically, *i-vector* is implemented as a generative methods. Conversely DL-based speaker embedding systems, *d-vector*, *x-vector*, and *t-vector* are rather discriminative. However, efforts have been made to convert discriminative speaker embedding systems into generative using *i-vectors* [64]. Consequently, present DL advancement has introduced a new generative strategy, particularly, generative adversarial network (GAN).

GAN [12] is a type of generative architecture that is observed to be widely implemented in speaker recognition tasks. Generally, GAN architecture contains a generator $G(\cdot)$ that generates data, and a discriminator $D(\cdot)$ that discriminates between actual data and data generated by $G(\cdot)$. Figure 13 illustrates the basic workflow of GAN. Mathematically, the objective of GAN can be defined as follows:

$$\min_G \max_D V_{GAN}(D, G) = \mathbb{E}_x[log(D(x))] + \mathbb{E}_z[log(1 - D(G(z)))] \quad (15)$$

The objective is to reduce the value of $\mathcal{L}(\cdot, \cdot)$, which is similar to a mini-max game. The $G(\cdot)$ tries to outbound the discriminator $D(\cdot)$, and vice-versa. $\mathbb{E}_x$ is the expected value of selecting x as an input from the actual data. In contrast, $\mathbb{E}_z$ is the expected value of selecting a noise z (from a noise distribution) to produce fake data from the generator.

GAN architectures are powerful enough to generate unseen data representations for a given condition [65]. Acquiring such diversity of data generation, GAN architectures have been successfully implemented for data augmentation strategies [66]–[68]. GAN architectures exploit interesting aspects to improve the speaker recognition systems architectures based on the dataset size, utterance size, and exploiting the perturbation of speaker recognition architectures. The following sections investigate some of the processes aforementioned.

1) SHORT UTTERANCE COMPENSATION

GAN has been used to extend short utterances into long speeches [69]. GAN has been observed to compensate for short utterances in the process, and the discriminator discriminates among fake and real utterances. The method implements a similar strategy to CGAN, where the generator translates short utterances into long instead of receiving random noises. Figure 14 explains the workflow of the proposed scheme. The objectives of the overall architecture are mathematically derived as follows:

$$\min_D loss = \mathbb{E}_x[log(D(x, y))] + \mathbb{E}_{x_s}[log(1 - D(G(x_s, y), y))] \quad (16)$$

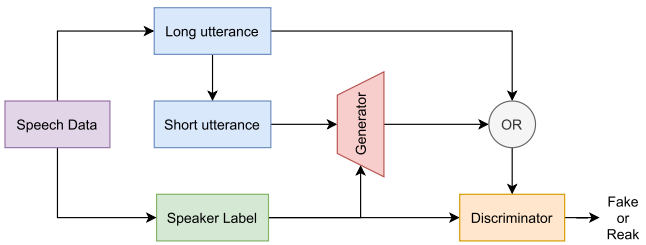

FIGURE 14. The image illustrates a general concept of converting short utterances into long using GAN [69]. The discriminator is fed actual long utterance or generator produced long utterance to discriminate. Both generator and discriminator receives the speaker/class information, similar to CGAN strategy.

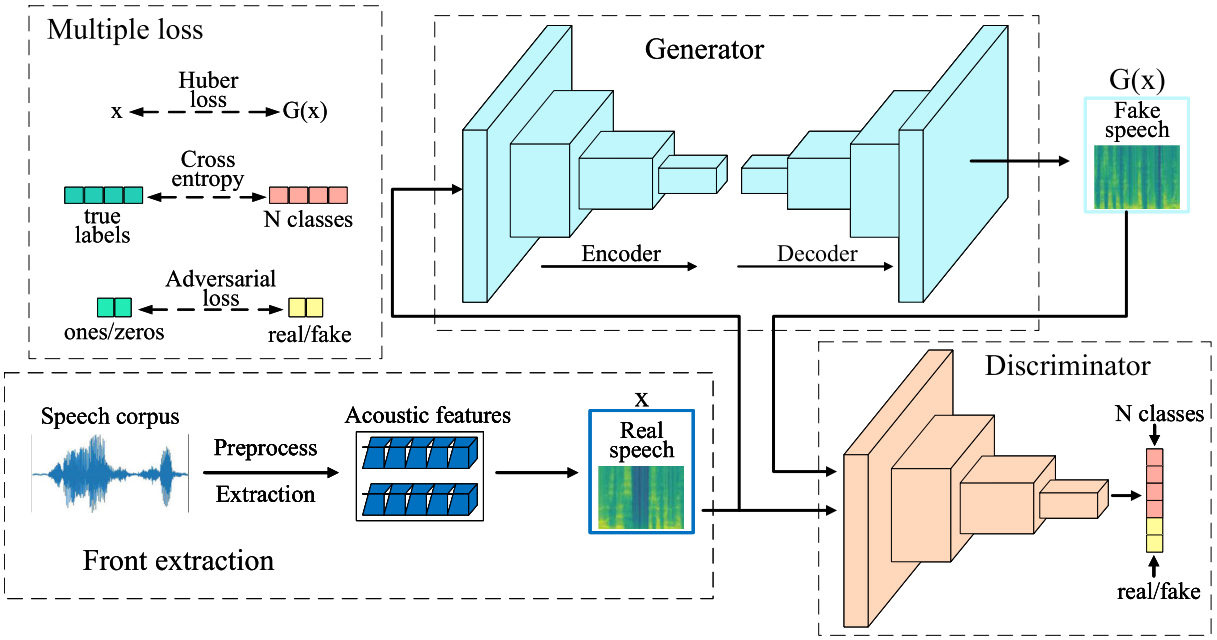

FIGURE 15. The figure illustrates the work of SpeakerGAN [70]. The training is similar to a GAN based augmentation strategy. The discriminator receives real and generator outputted speech features and classifies the speaker and real/fake data at once. Due to the feature shift of the generator, the discriminator receives diverse quality data. Hence, the discriminator does not overfit on lesser data and produces better accuracy.

$$\min_G loss = \mathbb{E}_{x_s}[(G(x_s, y) - x)^2] + E_{x_s}[log(1 - D(G(x_s, y), y))] \quad (17)$$

In the equation, x_s refers to the short utterances fed to the generator translating into long utterances. In the equation 16, the discriminator's loss remains similar to Conditional GAN's (CGAN) general strategy. However, the generator is trained based on the reconstruction loss, based on the mean-square difference between the target long utterance and generator generated long utterance. Also, a penalty is added based on the discriminator's precision.

To better describe the discriminator loss (equation 16), CGAN architecture is mathematically stated as follows:

$$\min_G \max_D V_{CGAN}(G, D) = \mathbb{E}_x[log(D(x, y))] + \mathbb{E}_z[log(1 - D(G(z, y), y))] \quad (18)$$

Here, y is the target speaker for the input speech x. Compared to GAN's general strategy (defined in equation 15), the discriminator CGAN specifically classifies the target class/speaker from the input, along with the discriminative information.

The compensation architecture directly translates short utterances into long and observed to increase the accuracy of *i-vector* systems to $[4 - 6]\%$. The authors also used the generator instead of the overall GAN architecture to compensate short utterances. Such implementation resulted in increasing the accuracy of *i-vectors* to only 2%. The overall implementation increased the proficiency of speaker recognition baselines for short speech segments.

2) SpeakerGAN: LEARNING FROM LESSER DATA

SpeakerGAN [70] is such a variant of CGAN trained on inadequate speech data (below 2 seconds per speaker) and achieved marginal performance. Compared to the CGAN architecture, the SpeakerGAN avoids passing speaker information y to the generator. The architecture is similar to CGAN and depends on the generator to generate a vast amount of diverse speech data. SpeakerGAN relies on the variation of data produced by the generator allowing the discriminator to recognize artificially augmented data, which improves the discriminator's robustness. After training, the discriminator is used as a speaker classifier. To facilitate the readers, the architecture is illustrated in Figure 15. The training procedure accumulates three loss functions. Cross-entropy loss is used to train the discriminator's classification process (derived in Equation 7). Huber loss is used to train the generator to generate an identical mapping of speech features related to the input. Huber loss can be defined as:

$$\mathcal{L}_H = \begin{cases} \frac{1}{2}(x - G(x))^2 & |y - G(x)| \leq \delta \\ \delta|x - G(x)| - \frac{\delta * 2}{2} & \text{otherwise} \end{cases} \quad (19)$$

Here, the authors used $\delta = 1$ in the loss implementation. Moreover, to train the overall adversarial network, a similar loss strategy of least square GAN [71] (LSGAN) is implemented. Mathematically LSGAN is implemented as follows:

$$\min_D V_{LSGAN}(D) = \frac{1}{2}E_x[(D(x) - 1)^2] + \frac{1}{2}E_z[(D(G(z)))^2] \quad (20)$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}E_z[((D(G(z))) - 1)^2] \quad (21)$$

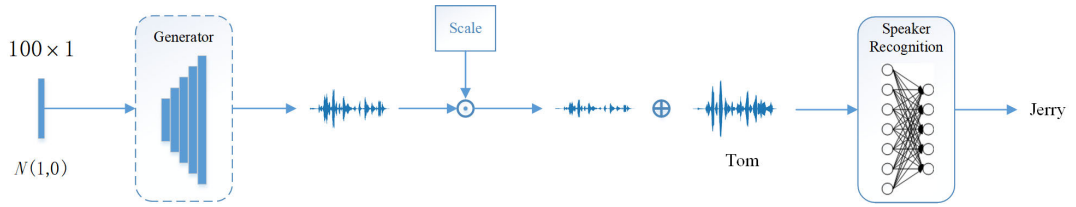

FIGURE 16. The figure depicts the strategy of the work [73]. A generator learns to produce a distracting audio sample. The sample is further scaled and merged with actual audio. A speaker recognition system mostly misclassifies the resulting audio.

The LSGAN's training objective properly instructs the discriminator gradients if the generator has not captured the speech data manifold [71]. The LSGAN is implemented based on the random noise data z. In contrast, the SpeakerGAN does not input noise data. Instead, it receives actual speaker data, and the objective is to mimic the data as much as possible. Hence, the architecture of the generator of SpeakerGan is similar to autoencoder architecture. Further, the generator architecture is implemented using gated linear units (GLU). The internal functionality of GLU layers is stated:

$$GLU(x) = (x * W + b) \circ \sigma(x * V + c) \tag{22}$$

Here $*$ defines a convolution operation, and $\circ$ represents pointwise/Hadamard multiplication. Further, $\sigma(\cdot)$ represents a sigmoid activation function. W, b and V, c are the weights and biases for the convolution and gate convolution respectively. The GLU layers are similar to a self-attention mechanism [72] which is promising. The overall implementation advantages diverse speech information extraction, resulting in better performance while trained with lesser data.

3) ADVERSARIAL PERTURBATIONS

Apart from increasing the speaker recognition framework's robustness, GAN has also been implemented in the speaker recognition systems to misclassify speakers [74], [75]. Such a technique is often referred to as perturbations. Further, a single noise input can cause false recognition of any speech input, and it is termed as universal perturbation [76], [77]. Adversarial attacks can be non-targeted and targeted. Non-targeted adversarial attack's task is to produce outputs that would be miss classified by a classifier. In contrast, targeted adversarial attacks deal with generating adversarial outputs classified as a specific target class by the classifier.

Perturbation of speaker recognition systems has also been observed using GAN architectures [73]. Figure 16 introduces an example of GAN based perturbation. The strategy includes a GAN that inputs a noise distribution and generates a distracting audio sample. The distracting audio sample is fused with an actual audio sample with a specific scale and fed to the classifier. The objective of the perturbation system can be stated as:

$$\max_{G} \mathcal{L}(G) \ = f(x + G(x)) \neq y \tag{23}$$

Here, $G(x)$ is the generator that generates disturbance, and x is the actual input. $f(\cdot)$ is a well-trained classifier. The authors [78] provide a wide range of investigation related to the topic and pointed out such a strategy can largely infect the speaker recognition methods. However, adversarial training can considerably defend such perturbation attacks [79].

D. META LEARNING IN SPEAKER RECOGNITION

Meta-learning [13] is a branch of DL dealing with classifying unknown or hardly known classes. Meta-learning is the current enchantment of DL methodologies because of its ability to learn from lesser data. Meta-learning methods learn to generalize on unseen data adequately. The generalization strategy is created using datasets that may or may not contain the query or test image classes. Such a learning scheme is attractive as, in most real-world problems, the amount of labelled data is scarce. Meta-learning approach includes two datasets, a support dataset $(x^s, y^s) \in \mathcal{S}$ and a training or enrollment dataset $(x, y) \in \mathcal{D}$. Based on the support dataset, methods are named K-shot C-class classification tasks. Here the total number of classes is C, and for each class, K number of samples are available. If $K = 0$, it is considered as a zero-shot learning strategy [80].

Meta-learning in DL methodology is still a converging research domain. Consequently, DL based meta-learning strategies are recently introduced in speaker recognition systems. Meta-learning is often classified into metric-based, model-based, and optimization-based approach. Further, these approaches have different categories and branches depending on the mechanisms. In Section V-D1, V-D2, and V-D3 we review some of the latest works falling in metric-based, model-based, and optimization-based meta-learning, respectively.

1) METRIC-BASED META-LEARNING

Metric-based DL architectures are mostly build using strategies like triplet loss architectures [81] (explained in Section IV-C), Siamese networks [34], Prototypical networks [82], etc. Metric based algorithms focus on increasing the distances between distinct classes while keeping similar classes closer. Such a strategy is greatly similar to speaker embedding's requirement. Metric learning emphasizes increasing the inter-class boundaries, which furnishes the generalization capability for unseen data. In the next section, prototypical

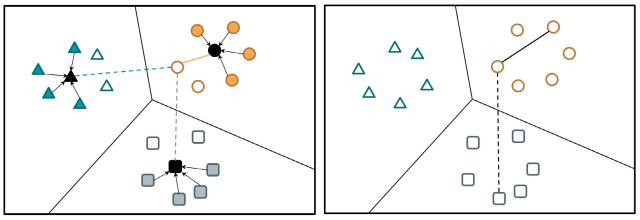

FIGURE 17. The figure illustrates the core difference between prototypical and triplet loss. The left image represents the prototypical loss strategy, which computes each classes' centroids and measures a data point's distance from class centroids (c_k). The right image represents the triplet loss strategy, which computes loss directly based on the distance between two data points. Triplet loss strategy might fail to centralize a cluster in lesser epoch and scarce data.

networks implemented on few-shot speaker recognition task is resolved.

a: PROTOTYPICAL NETWORK

Aside from triplet loss architectures, prototypical networks have also been observed in speaker recognition task [83]. Prototypical networks are trained based on prototypical loss, which solves significant issues of triplet loss architectures (defined in equation 4). Triplet loss architectures only consider increasing pairwise distances with a margin, ignoring each class's cluster centre. In contrast, prototypical loss contains a centroid c_k for each speaker in the enrollment dataset. Prototypical loss can be derived as:

$$c_k = \frac{1}{S_k} \sum_{(x_i, y_i) \in S_k} f(x_i^e) \tag{24}$$

$$\mathcal{L}_{PN} = \sum_{(x_i, y_i) \in \mathcal{D}} -log\ p(y_i = \hat{y}_i | x_i) = \frac{e^{-d(f(x_i), c_{y_i})}}{e^{\sum_k^C e^{-d(f(x_i), c_k)}}} \tag{25}$$

Here $\mathcal{L}_{PN}$ is the prototypical loss, $|S_k|$ is the number of speakers in the support dataset belonging to $k'th$ speaker. $d(\cdot, \cdot)$ is a distance parameter between two inputs. Prototypical loss is observed to perform better embeddings than triplet loss in the case of few-shot learning [83]. Triplet loss architectures can easily converge to a global minimum state while not keeping similar class embeddings into the same cluster. However, triplet loss architectures perform better if adequate data is available for speaker enrollment [84]. Hence, selection between triplet loss and prototypical loss greatly depends on data availability. Prototypical networks are also observed to better recognizing speakers from short utterances [85]. Prototypical networks trained with joined prototypical loss on enrollment and support dataset resemble better results.

2) MODEL-BASED META-LEARNING

The model-based meta-learning approach concentrates on faster learning of DL architectures with lesser data. The model-based approach often updates it's parameters rapidly, based on some pre-defined schemes. Further, some model-based approaches are perceived to be memory dependent. In the below sections, we describe some of the model-based strategies observed in the speaker recognition task.

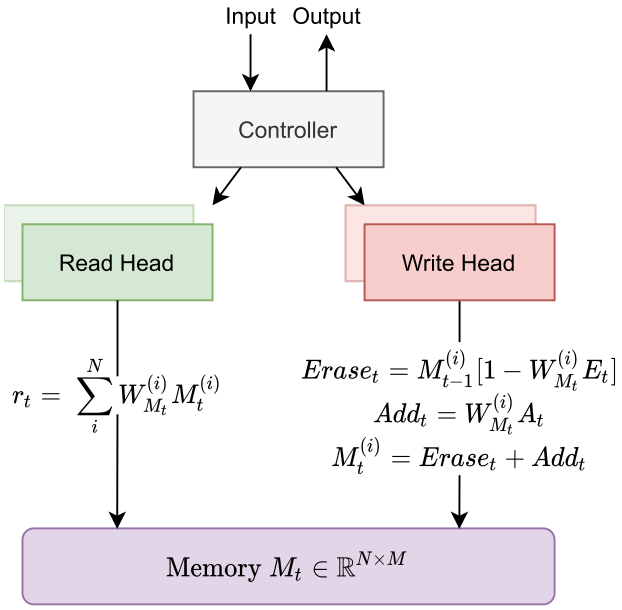

FIGURE 18. The figure clarifies a simplified process of neural turing machine. A controller receives input and encodes/downsamples the information. Further, it reads data from a memory/cache and saves encoded data if a new data pattern is observed. W_{M_t} is a weight vector used to focus on what to read from memory M_t in a timestep t. Further, it can erase information E_t and add new information A_t if required.

a: MEMORY AUGMENTED NEURAL NETWORK

The model-based approach specifically focuses on designing optimal models that can better recognize unseen entities. In speaker recognition, neural turing machine networks are being implemented. Neural turning machines [86] receive input, encode them, store them in memory, and search for a similar match from memory. The data storing mechanism is described as writing into memory, and reading data from memory is derived as reading from memory. A controller network maintains the overall read/write policy. Figure 18 contains a minimal brief of such a network.

Memory-augmented neural networks (MANN) [87], [88] is similar to a machine that uses memory, which has been observed to be applied in speaker diarization process [89]. The architectural constraints implement a relational memory core (RMC) [90] that controls memory modifications via attention method [91]. The architecture constructs a speaker memory $S = \{s_1, s_2, \ldots, s_n\}$ and the task is to find closest match to speaker profile s_i for a given speech input x. Each speaker profile s_i is a 2048-dimensional vector. The architecture implements *x-vectors* (discussed in Section IV-B) as a feature extractor, and further processing is done in RMC. The approach requires a heavy number of speech data and may cause overfit when data is limited.

3) OPTIMIZATION BASED META-LEARNING

Optimization-based meta-learning approaches focus on improving the gradient-based learning method. Generally, gradient-based optimization requires longer steps to converge. Moreover, gradient-based algorithms overfit in small

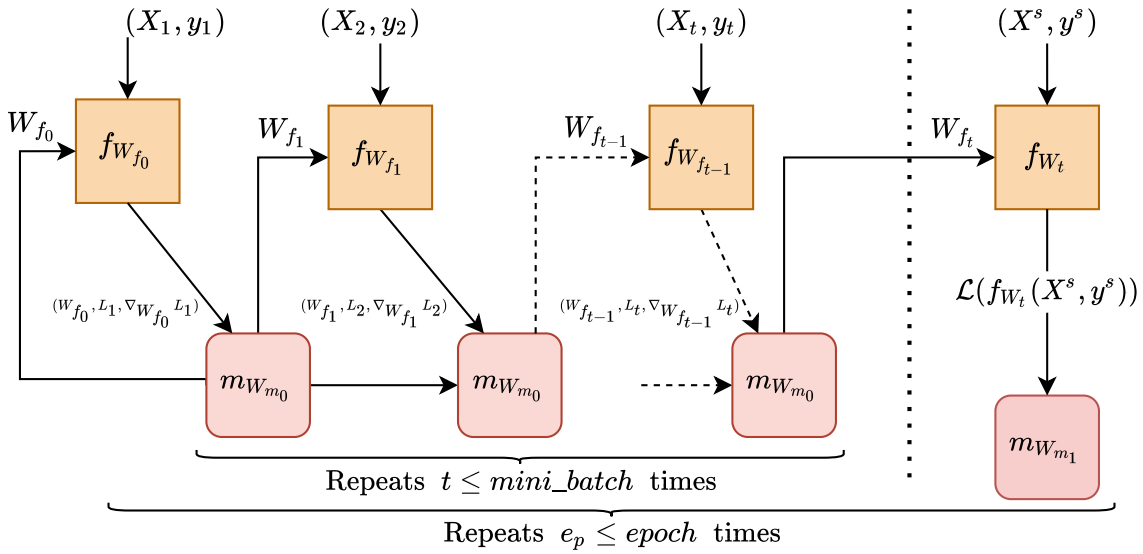

FIGURE 19. The figure illustrates the LSTM meta-learning approach. f_{W_t} is a learner/classifier and $m_{W_{m_t}}$ is a meta-learner at a specific time step t. The learner receives adjusted weights from meta-learner, computes output on enroll dataset minibatch $(X_i, y_i) \in \mathcal{D}$ and transfers it's current weight, loss value, and gradients (w.r.t. loss value) to the meta-learner. The meta-learner finds optimal weights for the specific case and again transfers the weights to the learner. The meta-learner's goal is to reduce loss on the enroll and support dataset $(X^s, y^s) \in \mathcal{S}$ jointly.

datasets. As the meta-learning approach deals with smaller datasets, general gradient-based strategies are neglected. In optimization-based procedures, the enrollment is conducted in smaller mini-batches compared to speaker recognition tasks' general batch size. The optimization problem's target is to find optimal weight W for a DL model $f_W(\cdot)$, that generates optimal results for all of the mini-batches and unseen. In the below sections, we expose some of the recent works conducted in the speaker recognition task based on the optimization methods.

a: LSTM META-LEARNING

LSTM meta-learning [92] approach finds the optimal adaptation of a DL model's weight enrolled on training dataset $(x, y) \in \mathcal{D}$ using a pair of LSTM layers. The pair of LSTM is named meta-learners. Such methods are also observed in speaker adaptation [93], where the objective is to learn from the enrollment dataset, acquiring better precision on the support dataset $(x^s, y^s) \in \mathcal{S}$. Meta-learner learns how the learner's (a DL classifier) weights W_f should be updated so that it achieves better precision on the small support dataset (x^s, y^s). The process of such method is formalized below:

$$v_t = \begin{pmatrix} W_{f_t} \\ L_t \\ \nabla_{W_{f_t}} L_t \end{pmatrix} \tag{26}$$

$$h_t = LSTM(v_t) \tag{27}$$

$$forget_t = \sigma(W_{forget} \cdot [h_t, f_{t-1}] + b_{forget}) \tag{28}$$

$$input_t = \sigma(W_{input} \cdot [h_t, i_t] + b_{input}) \tag{29}$$

$$W_{f_{t+1}} = forget_t \circ \theta_t - input_t \circ \nabla_{W_{f_t}} L \tag{30}$$

The meta-learner recieves a tuple of information: (a) current weight of the DL model, (b) loss value w.r.t. to (x, y), and (c) gradients of the loss w.r.t. to (x, y) (derived in Equation 26 and 27). The first lstm outputs a hidden representation h_t that is used by a handcrafted LSTM cell derived in equation 28 and 29. In the handcrafted LSTM cell, a forget gate $forget_t$ is calculated using the hidden representation h_t using weight W_{forget} and bias b_{forget} (equation 28). Further, an input gate $input_t$ calculates the rate of update (similar to learning rate) using weight W_{input} and bias b_{input} (equation 29). Finally, for each mini batch timestep t, the learner f_{W_t} is updated to $W_{f_{t+1}}$ as presented in equation 30.

The LSTM approach approximates the optimal converges from the enrollment dataset $\mathcal{D}$ such that it achieves superior performance in the support dataset $\mathcal{S}$. LSTM meta-learning approach is observed to gain better accuracy using a small portion of speaker data (speakers are presented in the test set), represented as a support dataset. Such procedures may solve the problem of requiring more data to recognize a particular system. However, LSTM based optimization has a memory limitation. The LSTM requires heavy speaker-dependent hidden weights $(v_t, forget_t, input_t)$, which often becomes memory inefficient for a large number of speakers.

b: MODEL AGNOSTIC META-LEARNING

Model agnostic meta-learning (MAML) [94], [95] is a simpler approach of meta-learning where the weights of the learner $f_w(\cdot)$ are updated based on the calculated loss on the enrollment dataset mini-batches (x, y). To explain MAML, the mathematical definition of MAML strategy is described:

$$W_t = W_{t-1} - \alpha \nabla_{W_{t-1}} m(W_{t-1}) \tag{31}$$

$$m(W) = \underset{W}{\operatorname{argmin}} \sum_{i}^{mini\_batch} \mathcal{L}(f_{W_i}(x), y) \tag{32}$$

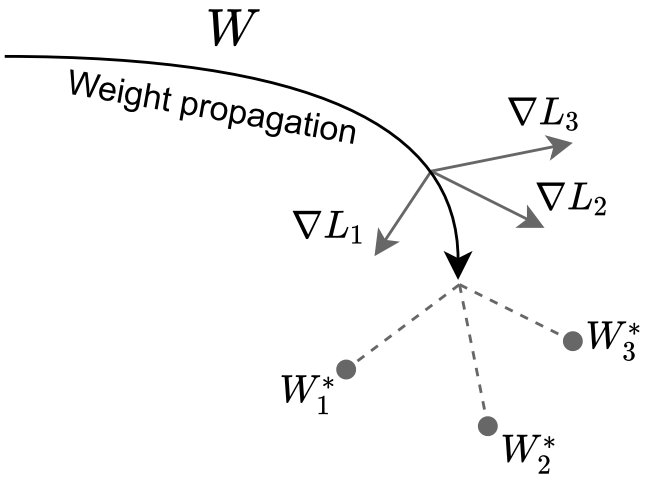

FIGURE 20. The figure illustrates the model agnostic meta-learning approach. The meta-learner receives information about the loss value $\{\nabla L_1, \nabla L_2, \ldots, \nabla L_n\}$ and weight for all mini-batches. After a complete iteration on the enrollment dataset, it updates the weights of the learner/classifier such that the updated weights produce a minimum loss for all of the mini-batches.

A meta-learning network tries to adapt the relationship of the loss and the learner's weights. The meta-learner suggests weights based on the overall enrollment dataset, such that the overall loss value reduces (equation 32). The meta-learner $m(W)$, suggests the weight updates of the learner model $f_W(\cdot)$, for a given weight W (equation 31). The meta-learner in equation 32 adjusts its weights via backpropagation, which is not derived. The MAML strategy is memory efficient as it does not depend on LSTM. LSTM based learning avoids over-learning for specific mini-batches, where MAML learns aggregately. Figure 20 illustrates a visual representation of the calibration process of MAML.

MAML strategy has been implemented in speaker adaptation task [96] to reduce the speaker variability between enrollment and testing phase. Although the MAML strategy has shown its diversity in various tasks [95], it hasn't produced marginal improvement in speaker adaptation. Such imbalance in speaker adaption requires more investigation related to speech feature extraction as well as DL-based solutions.

4) FEATURE BASED META LEARNING

a: META-ClusterGAN

Meta-ClusterGAN (MCGAN) [97] is a speaker clustering system specially designed for speaker diarization system. The architecture involves both the capability of meta-learning (prototypical loss) and GAN, achieving better speaker diarization clustering performance. Figure 21 introduces the training procedure of MCGAN, which is further elaborated in the current section.

The MCGAN contains a generator (G), encoder (E) and discriminator (D). The generator tries to mimic *x-vector* embeddings by receiving noise z_n and speaker label z_c. The discriminator discriminates between the actual *x-vector* input and generator generated input. The encoder tries to reconstruct z_c, and z_n from the generator generated embeddings. The training objective of the GAN is:

$$\min_{G,E} \max_{D} [w_1 \cdot U_{IWGAN}(D,G) + w_2 \cdot COS(G,E) + w_3 \cdot CE(G,E)] \tag{33}$$

The joint training criteria of MCGAN contains three weights w_1, w_2, w_3, which are tuned to prioritize specific objectives or loss. The $CE(G, E)$ calculates the cross-entropy loss (equation 7) between the generator's input and the encoders predicted class/speaker output. $COS(G, E)$ defines the cosine distance between the generator's noise input z_n and the encoder's predicted noise output z_n, derived as:

$$\min \mathcal{L}_{cos} = COS(G, E) = \mathbb{E}_z \left[1 - \frac{E(G(z_n)) \cdot z_n}{||E(G(z_n))||^2 \cdot ||z_n||^2} \right] \tag{34}$$

The MCGAN avoids general adversarial loss function, as it is observed to face mode collapse problem. Although Wasserstein GAN (WGAN) [98] solves the problem, the authors use an improved WGAN (IWGAN) [99] loss strategy that avoids such collapse issues. The $U_{IWGAN}(D, G)$ defines IWGAN loss derived as:

$$\begin{aligned} &\min_G \max_D U_{IWGAN}(D, G) \\ &\quad = \mathbb{E}_x[D(x)] - \mathbb{E}_z[D(G(z))] \\ &\qquad + \lambda \cdot \mathbb{E}_{x'}[(||\nabla_{x'} D(x')||^2 - 1)] \end{aligned} \tag{35}$$

$$x' = \epsilon x + (1 - \epsilon) \cdot G(z) \qquad \epsilon \in [0, 1] \tag{36}$$

In the equation, x' defines the accumulation of generator outputted and real vector embeddings scaled by random value $\epsilon \in [0, 1]$. The normalized gradient value of the discriminator network based on x' is also calculated and accumulated to the IWGAN loss value.

After the joint training of equation 33, the encoder is removed from the architecture. Further, the encoder is separately trained based on the input of *x-vector* and prototypical loss (derived in equation 24, 25). The training is conducted on small support set S. Hence the architecture requires a small amount of label data.

VI. SPEAKER RECOGNITION RESOURCES

The advancement of speaker recognition systems greatly depends on the programming tools available for speaker recognition investigation. Moreover, as DL is a data-driven approach, datasets play an essential role in the research and development of DL strategies. This section explores the programming tools and datasets available for speaker recognition systems.

A. PROGRAMMING TOOLKITS

The present advancement of DL-based speaker recognition architectures has widely been possible due to DL frameworks' rise. The frameworks give easy access to automatic differentiation [100], which are used to implement state-of-the-art DL strategies. Further, toolkits specially developed for speech processing have facilitated researchers implementing

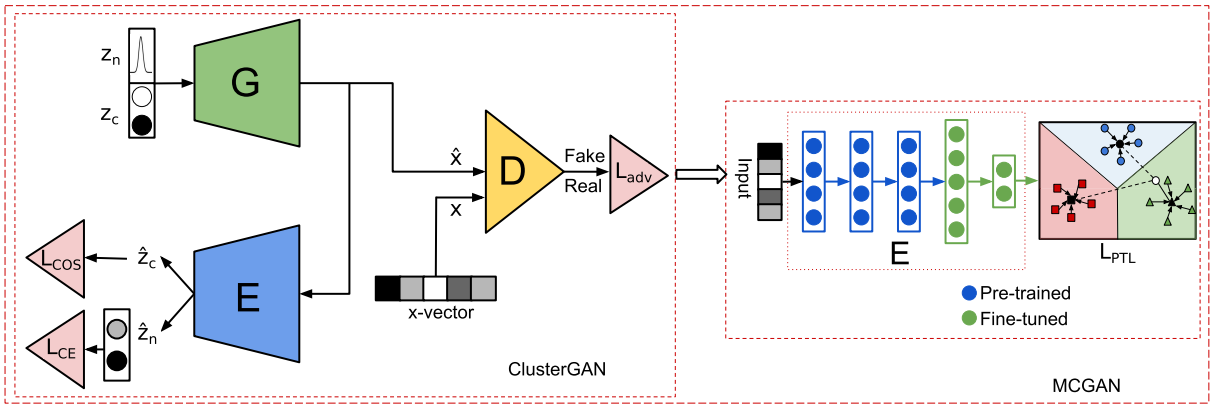

FIGURE 21. The MCGAN architecture includes a generator (G), encoder (E) and a discriminator (D). The generator receives noise z_n and speaker class z_c and tries to mimic x-vector embeddings. The encoder further tries to reconstruct z_n and z_c from the embedding. The encoder is further used as a cluster classifier. L_{cos}, L_{CE} is cosine-loss and cross-entropy loss between generator and encoder. L_{adv} is the adversarial loss between discriminator and encoder. L_{ptl} is prototypical loss of the final stage training of encoder, defined as MCGAN.

TABLE 4. The table illustrates the programming toolkits available for developing speaker recognition systems.

Name	Supported languages	Description
Librosa [105]	Python	Used for speech processing
Tensorflow [103]	Python, C++, JavaScript	Used for DL implementation
PyTorch [104]	Python	Used for DL implementation
Kaldi [101]	C++, Python	Speech feature extraction and acoustic modeling
Pytorch-Kaldi [106]	Python	A merge of Kaldi and PyTorch libraries
pyAudioAnalysis [107]	Python	Feature extraction, segmentation, and diarization tools
CMU Sphinx [108]	Java	Contains speech recognition models (obsolete)
ESPnet [109]	Python	End-to-end speech processing platform
HTK [102]	C	Hidden Markov model toolkit and models for speaker recognition

and investigating both existing and new speech and speaker recognition methods. Table 4 reports the available programming frameworks extensively used for speaker recognition, from pre-processing stage to the recognition stage.

Specifically for speaker recognition pre-processing, Kaldi [101] and HTK [102] has been one of the favorite choices among researchers. However, currently, Kaldi is mainly used for the pre-existing implementation of *i-vector* and *x-vector* systems. For DL based tasks, TensorFlow [103], and PyTorch [104] is being extensively used by researchers.

B. DATASETS

Apart from the toolkits, the advancement of speaker recognition systems also depends on the datasets available for research and development. Currently, an abundant amount of datasets are available for the evaluation of speaker recognition systems. Speaker recognition datasets contain data from various sources and criteria of data collection. Based on the origin and criteria, the datasets can be classified into the following segments:

- **Clean speech** is the general type of speaker recognition datasets collected from a quiet environment. Clean datasets are primarily collected in a studio environment. A conventional speaker recognition system refers to methods mostly trained and tested in clean datasets.
- **Telephony** speaker recognition datasets contain speech gathered using telephone and mobile devices. However, such collection can be done using two variations, directly collecting data from telephone mic or telephone calls.
- **Broadcast** speaker recognition dataset contains speech data from various physical broadcast devices: television, radio, etc. The broadcast dataset contains similar noise compared to the telephony dataset. However, it includes a variation due to sources.
- **Meeting** dataset contains data from a meeting environment. The speech utterances may overlap, may have laughter, noises and interruption. The dataset may slightly carry a far-field condition.
- **Far-field** dataset guarantees audio input devices are at a greater distance from the speaker. Such a condition is equivalent to in-home personal assistant devices, in which the device is placed far from the speaker. Further, far-field datasets may contain distracting noises, which is also challenging.
- **In-the-wild** is a unique condition for speaker recognition systems where the state is unknown. The speaker utterances may get overlapped, may also contain a far-field situation. Apart from containing heavy real-life noises, datasets may have meeting scenario as well.

Among the various dataset, the clean speech speaker recognition dataset only contains clean audio. In contrast, the rest of the categories may include noise, speech overlap, variance

TABLE 5. The table illustrates the present speaker recognition datasets, along with data distribution, collection source, and properties.

Name	Domain	Speakers	Utterances	Male-female distribution	Collection source	Properties
TIMIT [110]	Clean speech	630	6,300	70%-30%	Studio	Contains speaker information and transcripts at word and phoneme level. Best suited for speaker recognition and verification.
SWB [111]	Telephony	3114	33,039	56%-44%	Telephone recording	Contains speaker information and word-level transcripts. Best suited for speaker verification and recognition tasks.
ANDOSL [112]	Clean speech	204	33,900	50%-50%	Studio	Contains speaker information and transcripts in word and phoneme level. Best suited for speaker verification and recognition tasks.
POLYCOST [113]	Telephony	133	1,285	55%-45%	Telephone recording	Contains speaker information and transcripts. Best for speaker verification and recognition tasks.
ICSI Meeting Corpus [114]	Meeting	53	922	75%-25%	Multi-channel recording	Contains speaker information and word-level transcripts. Data is equivalent to in-the-wild constraints except for speaker emotion. Best suited for speaker diarization in noisy conditions.
ELSDSR [115]	Clean Speech	22	198	56%-44%	Multi-channel recording	Contains speaker information and transcripts. Best for exploring person-specific speaker characteristics.
MIT Mobile [116]	Telephony	88	7,884	54%-46%	Mobile recording	Contains shorter speech with transcripts. Best suited for speaker verification in noisy conditions with fewer data.
NIST SRE [117]	Clean and noisy Speech	2,000+	-	-	Muli-source	Contains speaker information in noisy and short duration data. Best for speaker recognition and verification in a noisy environment.
Forensic Comparison [118]	Telephone	552	1,264	-	Mobile and telephone recording	Contains speakers of different languages (Chinese, Spanish, English, Portuguese), having formal and informal speaking styles. Best for language-independent speaker recognition tasks.
MGB Challenge Dataset [119]	Broadcast Data	-	-	-	TV recording	Contains speaker information and transcripts. Suitable for speech to text translation, recognition, and diarization.
SITW [19]	In-the-wild	229	2,800	68%-32%	Various devices	Hand annotated in-the-wild dataset. Best for diarization and in-the-wild evaluation.
VoxCeleb [20]	In-the-wild	6112	1,128,246	61%-39%	Video and audio	Contains video audio and transcripts of youtube data. Best suited for in-the-wild evaluation.
VOiCES [120]	Far-field	300	-	50%-50%	Multi-source	Contains speaker information and transcripts. Best suited for far-field speaker recognition evaluation.
LibriSpeech [121]	Clean speech	1166+	-	52%-48%	Studio microphone	Contains speaker information and transcripts. Best suited for speech to text and speaker recognition tasks in clean and lightly noised environments.

of distance between speaker and input device, etc. Table 5 refers to the currently available datasets used for speaker recognition.

VII. CHALLENGES OF SPEAKER RECOGNITION SYSTEMS

Most speaker recognition architectures are implemented to solve a specific challenge. Hence, while solving one, architectures lack some aspects of speaker recognition standards. Table 6 overviews the performance of the architectures, based on the benchmark datasets. Table 7 illustrates the specialty of architectures, along with the drawbacks they face most. Further, some overall suggestions are also provided to improve the performance of the architectures based on accuracy and environmental perspectives. Most architectures are tested on a clean speech dataset. Hence they require noise cancellation strategies for real-world usage.

Based on the general implementations, the stagewise architectures and end-to-end architectures have both advantages and disadvantages. Stagewise architectures are losing their interest as the back-end layers are non-trainable. In contrast, end-to-end architectures also require pre-training in some phrases to get better weight initialization (this is always necessary for GAN architectures). However, end-to-end architectures can be trained jointly. Therefore the entire architecture

TABLE 6. The table illustrates the performance comparison of the discussed architectures based on the benchmark datasets (provided in Table 5).

Architecture	Task	Dataset	Implementation reference	Equal error rate (EER)	Detection cost function (DCF)	Classification error rate (CER)
i-vector [27]	Identification	NIST SRE [117]	[27]	9.68% ∼ 8.12%	0.574 ∼ 0.523	-
		SITW [19]	[27]	9.10% ∼ 6.09%	0.558 ∼ 0.472	-
i-vector + AutoEncoder [62]	Identification	NIST SRE [117]	[62]	4.50%	0.589	-
d-vector [26]	Verification	NIST SRE [117]	[26]	7.6% ∼ 2.9%	0.875 ∼ 0.539	-
x-vector [27]	Identification	NIST SRE [117]	[27]	8.0% ∼ 5.7%	0.632 ∼ 0.39	-
		SITW [19]	[27]	9.4% ∼ 4.16%	0.491 ∼ 0.391	-
t-vector [28]	Verification	NIST SRE [117]	[122]	11.0% ∼ 7.9%	-	-
		SWB [111]	[122]	3.58% ∼ 2.72%	-	-
RawNet [42]	Verification	VoxCeleb [20]	[42]	4.8% ∼ 4.0%	-	-
AM-MobileNet [49]	Identification	TIMIT [110]	[49]	-	-	0.43%
		MIT Mobile [116]	[49]	-	-	2.19%
SincNet [50]	Identification	TIMIT [110]	[49]	-	-	1.15%
		MIT Mobile [116]	[49]	-	-	1.27%
AM-SincNet [51]	Identification	TIMIT [110]	[49]	-	-	0.50%
		MIT Mobile [116]	[49]	-	-	0.52%
VLAD [51]	Verification	VoxCeleb [20]	[51]	8.03% ∼ 3.22%	-	-
Utterance compensation [69]	Verification	TIMIT [110]	[69]	8.73% ∼ 5.53%	0.531 ∼ 0.372	-
SpeakerGAN [70]	Identification	LibriSpeech [121]	[70]	2.80% ∼ 6.95%	-	-
Prototypical network [83]	Identification	VoxCeleb [20]	[83]	9.23% ∼ 8.17%	-	-

can jointly focus on feature extraction, speaker diversions, and classification of speakers based on the encoded deviations at the same time. Joint training is a considerable advantage for speaker recognition systems. As for neural networks, the process of recognition strategies is auto distributed into network layers through backpropagation. Also, for residual networks, some layers may get automatically unsued if necessary. Hence focusing on end-to-end architectures would result in better accuracy in the respective domain than stagewise implementations.

To explore the current challenges of speaker recognition systems, first, we need to define the properties of a novel speaker recognition system. A novel speaker recognition system is a theoretical system that is the most acceptable recognition system we want to achieve. Below we point out the properties of a novel speaker recognition system:

- **Precise:** Speaker recognition systems should generate excellent accuracy for recognizing speakers. Comparatively, the system should be more accurate than humans. Present DL based architectures, both end-to-end and stage-wise procedures are strongly focusing on speaker recognition precision. Yet, speaker recognition systems have the capacity for improvement due to the advancement of DL architectures. A more extensive architectural investigation is required.
- **Noise tolerant:** Speaker recognition systems should be tolerant to noises. Moreover, noises should not reduce the precision of the system. Although noise only refers to environmental factors, we also refer to the inconsistency of speech input due to different microphones, channel characteristics, and the field variance between speaker and input device as noise. The current speaker recognition systems target noise as an augmentation policy that improves DL models' feature selection [27]. Moreover, denoising autoencoders can be implemented as a dedicated noise filtering system [60].
- **Physically unbiased:** Speaker recognition systems should recognize speakers from various speech conditions, such as whisper [123] and emotion [124], [125] (laughter, weep, etc). Further, systems should not be biased over age and vocal diseases. As DL approaches are data-driven, such variation of data is generally unattainable. Hence, GAN-based augmentation methods can be applied to produce such variability of speaker features to improve the robustness over physical variations.

TABLE 7. The table illustrates the notable models discussed in the paper along with the strength, limitation and suggestions for improvement.

Architecture	Taxonomy	Strength	Limitation	Suggestion
d-vector [26]	Stagewise → front-end	Close performing network compared to *i-vector*.	Lower efficiency than *x-vector*.	Deeper architecture is required.
x-vector [27]	Stagewise → front-end	Better performing front-end model in variant situations.	Requires massive data and augmentation strategies in training.	VLAD-type temporal pooling should be introduced.
t-vector [28]	Stagewise → front-end	Can be easily integrated with any DL embedding architectures.	Not flawless compared to prototypical loss.	Better integrating DL architecture should be investigated.
DeepSpeaker [37]	End-to-end → residual	Average performing end-to-end speaker embedding generator.	Collapses while generating text-independent embeddings.	Both DL and loss strategy should be improved.
RawNet [42]	End-to-end → residual	Lightweight and directly processes raw-waveform.	Only suitable for speaker verification tasks.	Should be introduced for identification tasks, by modifying network strategy.
AM-MobileNet [49]	End-to-end → residual	Excellent for mobile devices by requiring lower parameters	Not competitive in all datasets. Noise-free data.	A denoising mechanism (such as DAE) can be introduced.
SincNet [50]	End-to-end → custom	Implementation of bandpass filters for recognizing speaker-dependent features	Noisy inputs can truncate faster convergence and reduce accuracy.	Denoising mechanism and better loss functions are required.
VLAD [41]	End-to-end → custom	Achieves comparatively better performance than temporal pooling layers.	Accuracy drops if the utterance length is reduced.	Speech compensation methods should be introduced.
AutoEncoder [62]	End-to-end → custom	Excellent for denoising speech inputs.	Requires a separate training process, similar to stagewise training.	Has possibility of merging VAD and denoising paralelly.
Utterance compensation [69]	End-to-end → GAN	Performs better on short-uttered speech.	Only tested on clean speech.	Compensation should be introduced for noisy input.
SpeakerGAN [70]	End-to-end → GAN	Requires small speech information.	Works for clean speech only.	'In-the-wild' scenario should be introduced.
Adversarial perturbation [73]	End-to-end → GAN	Can spoof a speaker recognition model with targeted and non-targeted attacks.	Requires recognition model and labeled data.	An unsupervised implementation is required.
Prototypical network [83]	End-to-end → meta	Requires low data and generates better result than triplet-loss.	Requires greater time to train model.	Prototypical loss dependent architectural investigation is required.
Memory augmented network [89]	End-to-end → meta	A memory search based meta-learning strategy depending on memorization.	Memory usage may increased based on the speaker population.	Should focus on memory compression for larger population.
LSTM meta-learning [93]	End-to-end → meta	Avoids overfitting on small datasets.	Higher memory usage.	Model agnostic methods should be implemented.
Model agnostic learning [96]	End-to-end → meta	Requires memory than LSTM meta-learning approach.	May not produce results better than LSTM meta-learning approach.	More speaker recognition based performance investigation is required.
Meta ClusterGAN [97]	End-to-end → meta End-to-end → GAN	Supports GAN-augmented data with lesser data requirement.	Comparative improvement of recognition is low.	More architectural investigation is required to improve the recognition benchmark.

- **Unconstrained recognition:** Speaker recognition system should be able to detect speakers even though the speech overlaps. In case of overlap, the system should be able to identify both speakers. Therefore, we search for a system that is aggregately best for diarization, identification and verification. Such cases are most similar to in-the-wild constraints. Hence, in-the-wild datasets should be extensively targeted to train and test DL architectures.
- **Minimal data:** Speaker recognition systems should focus on learning or adapt speakers from minimal data. However, we avoid approximating the quantity of data as unsupervised speaker recognition systems may not depend on data quantity. Meta-learning strategies can be implemented to reduce the requirement of long speech samples.
- **Tamper proof:** Speaker recognition systems should be tamper-proof against adversarial attacks and voice mimicry. In some cases, adversarial attacks require extensive investigation over the available architectures resulting in hard to generate [78]. However, voice mimicry does not significantly affect speaker recognition models [126]. Yet the challenge exists if a speakers speech is used from a playback device in a verification stage, the result may worsen [127], [128].

The present speaker recognition systems target almost all of the above constraints. However, they are biased to the general physical constraint of speech data. Nearly all current research works target clean speech, avoiding the natural changes of speech utterances due to emotional instabilities. Nevertheless, the current availability of large-scale in-the-wild datasets has encouraged researchers to develop robust architectures targeting a fraction of the derived challenges.

VIII. CONCLUSION

The paper presents an architectural investigation on deep learning based speaker recognition methods. It targets newcomers to acquaint themselves with the current progress of deep speaker recognition methods. Firstly, the paper instructs the general fundamentals of speaker recognition strategies. Then, it presents a taxonomy of deep speaker recognition architectures and explains some of the exceptional taxons' architectural perspectives. The paper further describes stage-wise architectures, residual networks, generative networks, custom modified networks, and meta-learning architectures in the progress of the investigation. Moreover, it summarizes the present programming frameworks available for implementing speaker recognition methods. The paper investigates the category of datasets available for speaker recognition systems along with a summarization. In what follows, the article explains the benefits and limitations of the described speaker recognition architectures along with recommendations. Finally, the manuscript introduces a novel speaker recognition system that points out the present speaker recognition challenges and motivates researchers towards implementing robust speaker recognition systems.

ACKNOWLEDGMENT

The authors would like to thank the Advanced Machine Learning (AML) Laboratory for resource sharing and precious opinions.

REFERENCES

[1] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018.

[2] Q. Le, L. Miralles-Pechuán, S. Kulkarni, J. Su, and O. Boydell, "An overview of deep learning in industry," *Data Analytics AI*, vol. 1, pp. 65–98, Aug. 2020.

[3] D. Sztahó, G. Szaszák, and A. Beke, "Deep learning methods in speaker recognition: A review," 2019, *arXiv:1911.06615*. [Online]. Available: http://arxiv.org/abs/1911.06615

[4] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," 2020, *arXiv:2001.00378*. [Online]. Available: http://arxiv.org/abs/2001.00378

[5] Z. Bai and X.-L. Zhang, "Speaker recognition based on deep learning: An overview," *Neural Netw.*, vol. 140, pp. 65–99, Aug. 2021.

[6] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. Interspeech*, Sep. 2018, pp. 1086–1090.

[7] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, Feb. 2018.

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[9] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[10] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.

[11] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. S. Rossi, "Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture," *IEEE Sensors J.*, vol. 21, no. 4, pp. 4827–4838, Feb. 2021.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 1–9.

[13] J. Vanschoren, "Meta-learning," in *Automated Machine Learning*. Cham, Switzerland: Springer, 2019, pp. 35–61.

[14] G. Dişken, Z. Tüfekçi, L. Saribulut, and U. Çevik, "A review on feature extraction for speaker recognition under degraded conditions," *IETE Tech. Rev.*, vol. 34, no. 3, pp. 321–332, May 2017.

[15] S. S. Tirumala, S. R. Shahamiri, A. S. Garhwal, and R. Wang, "Speaker identification features extraction methods: A systematic review," *Expert Syst. Appl.*, vol. 90, pp. 250–271, Dec. 2017.

[16] R. V. Pawar, R. M. Jalnekar, and J. S. Chitode, "Review of various stages in speaker recognition system, performance measures and recognition toolkits," *Anal. Integr. Circuits Signal Process.*, vol. 94, no. 2, pp. 247–257, Feb. 2018.

[17] F. Bahmaninezhad, C. Zhang, and J. H. L. Hansen, "An investigation of domain adaptation in speaker embedding space for speaker recognition," *Speech Commun.*, vol. 129, pp. 7–16, May 2021.

[18] L. Boves and E. den Os, "Speaker recognition in telecom applications," in *Proc. IEEE 4th Workshop Interact. Voice Technol. Telecommun. Appl. (IVTTA)*, Sep. 1998, pp. 203–208.

[19] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (SITW) speaker recognition database," in *Proc. Interspeech*, Sep. 2016, pp. 818–822.

[20] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Comput. Speech Lang.*, vol. 60, Mar. 2020, Art. no. 101027.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[22] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 4, pp. 788–798, May 2011.

[23] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey*, vol. 14, 2010, pp. 1–41.

[24] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 1695–1699.

[25] M. McLaren, Y. Lei, and L. Ferrer, "Advances in deep neural network approaches to speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 4814–4818.

[26] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 4052–4056.

[27] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5329–5333.

[28] C. Zhang, F. Bahmaninezhad, S. Ranjan, H. Dubey, W. Xia, and J. H. L. Hansen, "UTD-CRSS systems for 2018 NIST speaker recognition evaluation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 5776–5780.

[29] E. Barshan and P. Fieguth, "Stage-wise training: An improved feature learning strategy for deep models," in *Proc. Feature Extraction, Mod. Questions Challenges*, 2015, pp. 49–59.

[30] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1319–1327.

[31] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," 2017, *arXiv:1703.09507*. [Online]. Available: http://arxiv.org/abs/1703.09507

[32] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1–5.

[33] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[34] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, Lille, France, vol. 2, 2015, pp. 1–8.

[35] D. Garcia-Romero, A. McCree, S. Shum, N. Brummer, and C. Vaquero, "Unsupervised domain adaptation for I-vector speaker recognition," in *Proc. Odyssey, Speaker Lang. Recognit. Workshop*, vol. 8, 2014, pp. 1–5.

[36] Q. Wang, W. Rao, S. Sun, L. Xie, E. S. Chng, and H. Li, "Unsupervised domain adaptation via domain adversarial training for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 4889–4893.

[37] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," 2017, *arXiv:1705.02304*. [Online]. Available: http://arxiv.org/abs/1705.02304

[38] T. Glasmachers, "Limits of end-to-end learning," in *Proc. Asian Conf. Mach. Learn.*, 2017, pp. 17–32.

[39] Y.-Q. Yu, L. Fan, and W.-J. Li, "Ensemble additive margin softmax for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6046–6050.

[40] Y. Zhao, T. Zhou, Z. Chen, and J. Wu, "Improving deep CNN networks with long temporal context for text-independent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 6834–6838.

[41] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 5791–5795.

[42] J.-W. Jung, H.-S. Heo, J.-H. Kim, H.-J. Shim, and H.-J. Yu, "RawNet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," in *Proc. Interspeech*, Sep. 2019, pp. 1268–1272.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[44] H. Muckenhirn, M. Magimai.-Doss, and S. Marcell, "Towards directly modeling raw speech signal for speaker verification using CNNS," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 4884–4888.

[45] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 499–515.

[46] H.-S. Heo, J.-W. Jung, I.-H. Yang, S.-H. Yoon, H.-J. Shim, and H.-J. Yu, "End-to-end losses based on speaker basis vectors and all-speaker hard negative mining for speaker verification," in *Proc. Interspeech*, Sep. 2019, pp. 4035–4039.

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[49] J. A. C. Nunes, D. Macêdo, and C. Zanchettin, "AM-MobileNet1D: A portable model for speaker recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[50] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2018, pp. 1021–1028.

[51] J. A. C. Nunes, D. Macêdo, and C. Zanchettin, "Additive margin SincNet for speaker recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–5.

[52] H. Jégou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3304–3311.

[53] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[54] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5297–5307.

[55] Y. Zhong, R. Arandjelović, and A. Zisserman, "GhostVLAD for set-based face recognition," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2018, pp. 35–50.

[56] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

[57] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, and A. M. Dobaie, "Facial expression recognition via learning deep sparse autoencoders," *Neurocomputing*, vol. 273, pp. 643–649, Jan. 2018.

[58] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5810–5818.

[59] X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 1759–1763.

[60] Z. Zhang, L. Wang, A. Kai, T. Yamada, W. Li, and M. Iwahashi, "Deep neural network-based bottleneck feature and denoising autoencoder-based dereverberation for distant-talking speaker identification," *EURASIP J. Audio, Speech, Music Process.*, vol. 2015, no. 1, pp. 1–13, Dec. 2015.

[61] Q. Jin, T. Schultz, and A. Waibel, "Far-field speaker recognition," *IEEE Trans. Audio, Speech Language Process.*, vol. 15, no. 7, pp. 2023–2032, Sep. 2007.

[62] S. Shon, S. Mun, W. Kim, and H. Ko, "Autoencoder based domain adaptation for speaker recognition under insufficient channel information," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, 2017, pp. 1014–1018.

[63] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 14, 2002, p. 841.

[64] L. Xu, R. K. Das, E. Yilmaz, J. Yang, and H. Li, "Generative X-vectors for text-independent speaker verification," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2018, pp. 1014–1020.

[65] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2089–2093.

[66] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," 2017, *arXiv:1711.04340*. [Online]. Available: http://arxiv.org/abs/1711.04340

[67] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7278–7286.

[68] S. Wang, Y. Yang, Z. Wu, Y. Qian, and K. Yu, "Data augmentation using deep generative models for embedding based speaker recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 2598–2609, 2020.

[69] Z. Hu, Y. Fu, Y. Luo, X. Xu, Z. Xia, and H. Zhang, "Speaker recognition based on short utterance compensation method of generative adversarial networks," *Int. J. Speech Technol.*, vol. 23, no. 2, pp. 443–450, Jun. 2020.

[70] L. Chen, Y. Liu, W. Xiao, Y. Wang, and H. Xie, "SpeakerGAN: Speaker identification with conditional generative adversarial network," *Neurocomputing*, vol. 418, pp. 211–220, Dec. 2020.

[71] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2794–2802.

[72] G. Tang, M. Müller, A. Rios, and R. Sennrich, "Why self-attention? A targeted evaluation of neural machine translation architectures," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 4263–4272.

[73] J. Li, X. Zhang, C. Jia, J. Xu, L. Zhang, Y. Wang, S. Ma, and W. Gao, "Universal adversarial perturbations generative network for speaker recognition," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2020, pp. 1–6.

[74] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4422–4431.
[75] S. Hussain, P. Neekhara, M. Jere, F. Koushanfar, and J. McAuley, "Adversarial deepfakes: Evaluating vulnerability of deepfake detectors to adversarial examples," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3348–3357.
[76] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.
[77] P. Neekhara, S. Hussain, P. Pandey, S. Dubnov, J. McAuley, and F. Koushanfar, "Universal adversarial perturbations for speech recognition systems," in *Proc. Interspeech*, Sep. 2019, pp. 481–485.
[78] A. Jati, C.-C. Hsu, M. Pal, R. Peri, W. Abdalmageed, and S. Narayanan, "Adversarial attack and defense strategies for deep speaker recognition systems," *Comput. Speech Lang.*, vol. 68, Jul. 2021, Art. no. 101199.
[79] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–28.
[80] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning—The good, the bad and the ugly," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4582–4591.
[81] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Proc. Int. Workshop Similarity-Based Pattern Recognit.* Cham, Switzerland: Springer, 2015, pp. 84–92.
[82] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4080–4090.
[83] J. Wang, K.-C. Wang, M. T. Law, F. Rudzicz, and M. Brudno, "Centroid-based deep metric learning for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3652–3656.
[84] M. J. Lee and J. So, "Metric-based learning for nearest-neighbor few-shot image classification," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2021, pp. 460–464.
[85] S. M. Kye, Y. Jung, H. B. Lee, S. J. Hwang, and H.-R. Kim, "Meta-learning for short utterance speaker recognition with imbalance length pairs," in *Proc. Interspeech*, 2020, pp. 2982–2986.
[86] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," 2014, *arXiv:1410.5401*. [Online]. Available: http://arxiv.org/abs/1410.5401
[87] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2014, *arXiv:1410.3916*. [Online]. Available: http://arxiv.org/abs/1410.3916
[88] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1842–1850.
[89] N. Flemotomos and D. Dimitriadis, "A memory augmented architecture for continuous speaker identification in meetings," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 6524–6528.
[90] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap, "Relational recurrent neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7310–7321.
[91] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[92] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017.
[93] O. Klejch, J. Fainberg, and P. Bell, "Learning to adapt: A meta-learning approach for speaker adaptation," in *Proc. Interspeech*, Sep. 2018, pp. 867–871.
[94] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Proc. NIPS*, 2016, pp. 1–17.
[95] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[96] O. Klejch, J. Fainberg, P. Bell, and S. Renals, "Speaker adaptive training using model agnostic meta-learning," in *Proc. IEEE Autom. Speech Recognit. Understand. Workshop (ASRU)*, Dec. 2019, pp. 881–888.
[97] M. Pal, M. Kumar, R. Peri, T. J. Park, S. H. Kim, C. Lord, S. Bishop, and S. Narayanan, "Meta-learning with latent space clustering in generative adversarial network for speaker diarization," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 1204–1219, 2021.
[98] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[99] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5769–5779.
[100] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, Apr. 2018.
[101] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop Automat. Speech Recognit. Understand.*, Jan. 2011, pp. 1–4.
[102] S. J. Young and S. Young, "The HTK hidden Markov model toolkit: Design and philosophy," Entropic Cambridge Res. Lab., 1994, pp. 2–44, vol. 2.
[103] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: http://arxiv.org/abs/1603.04467
[104] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lere, "Automatic differentiation in pytorch," in *Proc. NIPS Autodiff Workshop*, Long Beach, CA, USA, Dec. 2017.
[105] B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in Python," in *Proc. 14th Python Sci. Conf.*, vol. 8, 2015, pp. 18–25.
[106] M. Ravanelli, T. Parcollet, and Y. Bengio, "The PyTorch-Kaldi speech recognition toolkit," in *Proc. ICASSP*, 2019, pp. 6465–6469.
[107] T. Giannakopoulos, "PyAudioAnalysis: An open-source Python library for audio signal analysis," *PLoS ONE*, vol. 10, no. 12, Dec. 2015, Art. no. e0144610.
[108] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, "The CMU Sphinx-4 speech recognition system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Hong Kong, vol. 1, Dec. 2003, pp. 2–5.
[109] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," *Proc. Interspeech*, 2018, pp. 2207–2211.
[110] W. M. Fisher, "Ther DARPA speech recognition research database: Specifications and status," in *Proc. DARPA Workshop Speech Recognit.*, Feb. 1986, pp. 93–99.
[111] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Mar. 1992, pp. 517–520.
[112] J. B. Millar, J. P. Vonwiller, J. M. Harrington, and P. J. Dermody, "The Australian national database of spoken language," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 1, Apr. 1994, p. I-97.
[113] J. Hennebert, H. Melin, D. Petrovska, and D. Genoud, "POLYCOST: A telephone-speech database for speaker recognition," *Speech Commun.*, vol. 31, nos. 2–3, pp. 265–270, Jun. 2000.
[114] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters, "The ICSI meeting corpus," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1, Apr. 2003, p. 1.
[115] L. Feng and L. K. Hansen, "A new database for speaker recognition," Inform. Math. Model., Tech. Univ. Denmark, DTU, Lyngby, Denmark, IMM-Tech. Rep. 2005-05, 2005. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3662-full.html
[116] R. Woo, A. Park, and T. Hazen, "The MIT mobile device speaker verification corpus: Data collection and preliminary experiments," in *Proc. IEEE Odyssey-Speaker Lang. Recognit. Workshop*, Jun. 2006, pp. 1–6.
[117] C. S. Greenberg, V. M. Stanford, A. F. Martin, M. Yadagiri, G. R. Doddington, J. J. Godfrey, and J. Hernandez-Cordero, "The 2012 NIST speaker recognition evaluation," in *Proc. INTERSPEECH*, 2013, pp. 1971–1975.
[118] G. Morrison, C. Zhang, E. Enzinger, F. Ochoa, D. Bleach, M. Johnson, B. Folkes, S. De Souza, N. Cummins, and D. Chow. (2015). *Forensic Database of Voice Recordings of 500+ Australian English Speakers*. [Online]. Available: http://databases.forensic-voice-comparison.net.

[119] P. Bell, M. J. F. Gales, T. Hain, J. Kilgour, P. Lanchantin, X. Liu, A. McParland, S. Renals, O. Saz, M. Wester, and P. C. Woodland, "The MGB challenge: Evaluating multi-genre broadcast media recognition," in *Proc. IEEE Workshop Automat. Speech Recognit. Understand. (ASRU)*, Dec. 2015, pp. 687–693.

[120] C. Richey, M. A. Barrios, Z. Armstrong, C. Bartels, H. Franco, M. Graciarena, A. Lawson, M. K. Nandwana, A. Stauffer, J. van Hout, P. Gamble, J. Hetherly, C. Stephenson, and K. Ni, "Voices obscured in complex environmental settings (VOICES) corpus," 2018, *arXiv:1804.05053*. [Online]. Available: http://arxiv.org/abs/1804.05053

[121] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[122] C. Zhang, K. Koishida, and J. H. L. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 9, pp. 1633–1644, Sep. 2018.

[123] V. Vestman, D. Gowda, M. Sahidullah, P. Alku, and T. Kinnunen, "Speaker recognition from whispered speech: A tutorial survey and an application of time-varying linear prediction," *Speech Commun.*, vol. 99, pp. 62–79, May 2018.

[124] B. D. Sarma and R. K. Das, "Emotion invariant speaker embeddings for speaker identification with emotional speech," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, 2020, pp. 610–615.

[125] S. G. Koolagudi, S. E. Fatima, and K. S. Rao, "Speaker recognition in the case of emotional environment using transformation of speech features," in *Proc. CUBE Int. Inf. Technol. Conf.*, 2012, pp. 118–123.

[126] V. Vestman, T. Kinnunen, R. G. Hautamäki, and M. Sahidullah, "Voice mimicry attacks assisted by automatic speaker verification," *Comput. Speech Lang.*, vol. 59, pp. 36–54, Jan. 2020.

[127] F. Alegre, A. Janicki, and N. Evans, "Re-assessing the threat of replay spoofing attacks against automatic speaker verification," in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, 2014, pp. 1–6.

[128] T. Kinnunen, N. Evans, J. Yamagishi, K. A. Lee, M. Sahidullah, M. Todisco, and H. Delgado, "Asvspoof 2017: Automatic speaker verification spoofing and countermeasures challenge evaluation plan," *Training*, vol. 10, no. 1508, p. 1508, 2017.

ABU QUWSAR OHI received the degree in computer science. He is currently working as a Lecturer and a Research Assistant with the Department of Computer Science and Engineering, Bangladesh University of Business and Technology. He is proficient in prominent frameworks, including TensorFlow, Keras, NumPy, and Matplotlib. He is also passionate about learning fundamental algorithms, including data structures, dynamic programming, and game theory. He has experience working in famous languages, including Python and C++. His research interests include deep learning algorithms, pattern recognition, computer vision, and reinforcement learning. Apart from his research works, he is a Competitive Programmer and has attained more than ten Bangladeshi national programming contests, including National Collegiate Programming Contest (NCPC) and International Collegiate Programming Contest (ICPC).

M. F. MRIDHA (Senior Member, IEEE) received the Ph.D. degree in AI/ML from Jahangirnagar University, in 2017. In June 2007, he joined as a Lecturer with the Department of Computer Science and Engineering, Stamford University Bangladesh. He was promoted as a Senior Lecturer with the Department of Computer Science and Engineering, Stamford University Bangladesh, in October 2010, and promoted as an Assistant Professor, in October 2011. In May 2012, he joined the University of Asia Pacific (UAP) as an Assistant Professor. From 2012 to 2019, he worked as a Faculty Member with the Department of Computer Science and Engineering (CSE), UAP, and a Graduate Coordinator. He is currently working as an Associate Professor with the Department of Computer Science and Engineering, Bangladesh University of Business and Technology. For more than ten years, he has been with the master's and undergraduate students as a Supervisor of their thesis work. His research experience, within both academia and industry, results in over 80 journal articles and conference papers. His research interests include artificial intelligence (AI), machine learning, deep learning, natural language processing (NLP), and big data analysis. He has served as a program committee member for several international conferences/workshops. He served as an associate editor for several journals.

MD. ABDUL HAMID was born in Sonatola, Pabna, Bangladesh. He received the higher secondary degree from the Rajshahi Cadet College, Bangladesh, in 1995, the B.E. degree in computer and information engineering from International Islamic University Malaysia (IIUM), in 2001, and the joint master's and Ph.D. degrees in information communication from the Department of Computer Engineering, Kyung Hee University, South Korea, in August 2009. His education life spans over different countries in the world. He has been in the teaching profession throughout his life, which also spans over different parts of the globe. From 2002 to 2004, he was a Lecturer with the Department of Computer Science and Engineering, Asian University of Bangladesh, Dhaka. From 2009 to 2012, he was an Assistant Professor with the Department of Information and Communications Engineering, Hankuk University of Foreign Studies (HUFS), South Korea. From 2012 to 2013, he was an Assistant Professor with the Department of Computer Science and Engineering, Green University of Bangladesh. From 2013 to 2016, he was an Assistant Professor with the Department of Computer Engineering, Taibah University, Medina, Saudi Arabia. From 2016 to 2017, he was an Associate Professor with the Department of Computer Science, Faculty of Science and Information Technology, American International University-Bangladesh, Dhaka. From 2017 to 2019, he was an Associate Professor and a Professor with the Department of Computer Science and Engineering, University of Asia Pacific, Dhaka. Since 2019, he has been a Professor with the Department of Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. His research interests include network/cyber-security, natural language processing, machine learning, wireless communications, and networking protocols.

MUHAMMAD MOSTAFA MONOWAR (Member, IEEE) received the B.Sc. degree in computer science and information technology from the Islamic University of Technology (IUT), Bangladesh, in 2003, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2011. He worked as a Faculty Member with the Department of Computer Science and Engineering, University of Chittagong, Bangladesh. He is currently working as an Associate Professor with the Department of Information Technology, King Abdulaziz University, Saudi Arabia. His research interests include wireless networks, mostly *ad-hoc*, sensor, and mesh networks, including routing protocols, MAC mechanisms, IP, and transport layer issues, cross-layer design, and QoS provisioning; security and privacy issues; natural language processing. He has served as a program committee member for several international conferences/workshops. He served as an editor for a couple of books published by CRC Press and Taylor & Francis Group. He also served as a guest editor for several journals.