

SYSTEM AND METHOD FOR PERSONALIZED SPEAKER VERIFICATION**CROSS REFERENCE TO RELATED APPLICATION**

[1] The present application is based on and claims priority to the International Patent Application No. PCT/CN2019/114812, filed on October 31, 2019, and titled "SYSTEM AND METHOD FOR DETERMINING VOICE CHARACTERISTICS", the entire contents of which are incorporated herein by reference in the entirety.

TECHNICAL FIELD

[2] This application generally relates to systems and methods for personalized speaker verification.

BACKGROUND

[3] Many applications have been developed based on human voice characteristics. Speaker verification may authenticate speaker identity given enrolled acoustic characteristics of the speaker and some speech utterances as trials, and output a binary decision of whether accepting or rejecting the speech utterances as associated with the speaker. For example, individual identities may be linked to their unique voices so that users can be verified according to their voices. To that end, machine learning algorithms such as deep learning have been proposed to train computer systems to recognize human voices. Deep learning, also known as deep neural network (DNN), is a subset of machine learning in artificial intelligence that has networks capable of learning from data that is unstructured (which can be labeled or unlabeled).

[4] The traditional speaker verification system uses a general speaker recognition model for all users (one-for-all) without any personalized update for target user, and hence lacks robustness and flexibility. Specifically, the traditional machine learning model often fixes its parameters after training with data samples, and the products carrying the models are offered to users contain the same fixed parameters. Therefore, it is desirable to provide machine learning models that are personalized for individuals.

SUMMARY

[5] Various embodiments of the specification include, but are not limited to, systems, methods, and non-transitory computer readable media for personalized speaker verification.

[6] According to some embodiments, a computer-implemented method for personalized speaker verification comprises: obtaining first speech data of a speaker as a positive sample and second speech data of an entity different from the speaker as a negative sample; feeding the positive

sample and the negative sample to a first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker; obtaining a gradient based at least on the positive voice characteristic and the negative voice characteristic; and feeding the gradient to the first model to update one or more parameters of the first model to obtain a second model for personalized speaker verification.

[7] In some embodiments, obtaining the first speech data of the speaker as the positive sample and the second speech data of the entity different from the speaker as the negative sample comprises: obtaining one or more speech segments of the speaker as the positive sample and one or more speech segments of one or more people other than the speaker as the negative sample. Feeding the positive sample and the negative sample to the first model to correspondingly output the positive voice characteristic and the negative voice characteristic of the speaker comprises: feeding the one or more speech segments of the speaker into the first model to correspondingly output one or more positive sample vectors, and feeding the one or more speech segments of the one or more people other than the speaker into the first model to correspondingly output one or more negative sample vectors.

[8] In some embodiments, the method further comprises averaging the one or more positive sample vectors to obtain a template vector of the speaker.

[9] In some embodiments, the method further comprises: obtaining speech data of a user; feeding the obtained speech data to the second model to obtain an input vector of the user; comparing the input vector of the user with the template vector of the speaker; and verifying if the user is the speaker based at least on the comparison.

[10] In some embodiments, obtaining the gradient based at least on the positive voice characteristic and the negative voice characteristic comprises: feeding the one or more positive sample vectors and the one or more negative sample vectors into a neural network classifier to obtain one or more gradient vectors.

[11] In some embodiments, obtaining the gradient based at least on the positive voice characteristic and the negative voice characteristic further comprises: averaging the one or more gradient vectors to obtain an average gradient vector of the speaker as the gradient.

[12] In some embodiments, feeding the gradient to the first model to update the one or more parameters of the first model comprises: feeding the average gradient vector of the speaker to the

first model to update the one or more parameters of the first model; and the one or more parameters associate different neural layers of the first model.

[13] In some embodiments, feeding the one or more positive sample vectors and the one or more negative sample vectors into the neural network classifier to obtain one or more gradient vectors comprises: obtaining the gradient based at least on backward propagation through a cross-entropy loss function of the neural network classifier.

[14] In some embodiments, feeding the gradient to the first model to update the one or more parameters of the first model comprises: feeding the gradient to the first model to update the one or more parameters of the first model based at least on the gradient and an online machine learning rate.

[15] In some embodiments, feeding the gradient to the first model to update the one or more parameters of the first model based at least on the gradient and the online machine learning rate comprises: updating the one or more parameters in a direction in which the gradient descends at a fastest online machine learning rate.

[16] In some embodiments, before feeding the positive sample and the negative sample to the first model for determining voice characteristics, the first model has been trained at least by jointly minimizing a first loss function that optimizes speaker classification and a second loss function that optimizes speaker clustering.

[17] In some embodiments, the first loss function comprises a non-sampling-based loss function; and the second function comprises a Gaussian mixture loss function with non-unit multi-variant covariance matrix.

[18] According to other embodiments, a system for personalized speaker verification comprises one or more processors and one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to perform the method of any of the preceding embodiments.

[19] According to yet other embodiments, a non-transitory computer-readable storage medium is configured with instructions executable by one or more processors to cause the one or more processors to perform the method of any of the preceding embodiments.

[20] According to still other embodiments, an apparatus for personalized speaker verification comprises a plurality of modules for performing the method of any of the preceding embodiments.

[21] According to some embodiments, a system for personalized speaker verification comprises one or more processors and one or more non-transitory computer readable storage media storing instructions executable by the one or more processors to cause the one or more processors to perform operations comprising: obtaining first speech data of a speaker as a positive sample and second speech data of an entity different from the speaker as a negative sample; feeding the positive sample and the negative sample to a first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker; obtaining a gradient based at least on the positive voice characteristic and the negative voice characteristic; and feeding the gradient to the first model to update one or more parameters of the first model to obtain a second model for personalized speaker verification.

[22] According to other embodiments, a non-transitory computer-readable storage medium is configured with instructions executable by one or more processors to cause the one or more processors to perform operations comprising: obtaining first speech data of a speaker as a positive sample and second speech data of an entity different from the speaker as a negative sample; feeding the positive sample and the negative sample to a first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker; obtaining a gradient based at least on the positive voice characteristic and the negative voice characteristic; and feeding the gradient to the first model to update one or more parameters of the first model to obtain a second model for personalized speaker verification.

[23] According to yet other embodiments, an apparatus for personalized speaker verification comprises a first obtaining module for obtaining first speech data of a speaker as a positive sample and second speech data of an entity different from the speaker as a negative sample; a first feeding module for feeding the positive sample and the negative sample to a first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker; a second obtaining module for obtaining a gradient based at least on the positive voice characteristic and the negative voice characteristic; and a second feeding module for feeding the gradient to the first model to update one or more parameters of the first model to obtain a second model for personalized speaker verification.

[24] Embodiments disclosed herein have one or more technical effects. In some embodiments, the personalized speaker verification model for a speaker is personalized by training model

parameters with at least the speaker's speech data. This strengthens the model's robustness and flexibility and offers accurate identity verification. In some embodiments, an online machine learning strategy is used to train the personalized speaker verification model, which helps to improve the model's performance. In some embodiments, both positive sample (e.g., speech data of a target speaker for personalizing speaker verification) and negative sample (e.g., speech data not uttered by the speaker) are used for model training. This improves the robustness and accuracy of the personalized speaker verification model. In one embodiment, the positive and negative samples are fed into a first model (referred to as "a first model for determining voice characteristics") to obtain a template vector and a gradient vector, both of which may be stored in association with the speaker. In some embodiments, the gradient vector is fed back to the first model to update the first model and obtain the updated first model as a second model (referred to as "a second model for personalized speaker verification"). The second model may contain personalized parameters for each target speaker, and may be used for personalized speaker verification.

[25] These and other features of the systems, methods, and non-transitory computer readable media disclosed herein, as well as the methods of operation and functions of the related elements of structure and the combination of parts and economies of manufacture, will become more apparent upon consideration of the following description and the appended claims with reference to the accompanying drawings, all of which form a part of this specification, wherein like reference numerals designate corresponding parts in the various figures. It is to be expressly understood, however, that the drawings are for purposes of illustration and description only and are not intended as limiting.

BRIEF DESCRIPTION OF THE DRAWINGS

[26] FIG. 1 illustrates a system for personalized speaker verification in accordance with some embodiments.

[27] FIG. 2A illustrates a method for obtaining feature vectors in accordance with some embodiments.

[28] FIG. 2B illustrates a method for training an example of a first model for determining voice characteristics in accordance with some embodiments.

[29] FIG. 2C illustrates a method for personalized speaker verification in accordance with some embodiments.

[30] FIG. 3 illustrates a method for personalized speaker verification in accordance with some embodiments.

[31] FIG. 4 illustrates a block diagram of a computer system for personalized speaker verification in accordance with some embodiments.

[32] FIG. 5 illustrates a block diagram of a computer system in which any of the embodiments described herein may be implemented.

DETAILED DESCRIPTION

[33] Voice characteristics as personal traits have potential for various applications. Voice characteristics may include one or more of: volume, pitch, intonation (rising or falling pitch), tone, duration, speech rate, loudness, and the like. Each person may have one or more unique voice characteristics to distinguish from other people. In one example, speaker verification may authenticate speaker identity given enrolled voice characteristics of the speaker and some speech utterances as trials. Speaker verification outputs a binary decision of acceptance or rejection of unidentified speech utterances as associated with the speaker.

[34] Voice characteristics may or may not be speech-based. Thus, speaker verification systems can be further categorized as text-independent or text-dependent. A text-independent system does not fix content of the utterances to some lexical phrases, in contrast to a text-dependent system. In one example, for a text-dependent system, all users have to utter the same preset phrase to be recognized based on their voices; but for a text-independent system, the users can utter different phrases or voices and still be recognized.

[35] Various methods have been proposed to obtain real-valued, compact, and low-dimensional vectors to represent speaker characteristics using deep learning. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning from data that is unstructured or unlabeled. Deep learning can be supervised, semi-supervised, or unsupervised. Recent works attempt to incorporate a variety of loss functions, such as triplet loss, center loss, and contrastive loss to train speaker embeddings (a set of high-level feature representations through deep learning). Speaker embedding may refer to a learned embedding that captures segment-level acoustic representation from variable-length speech segments to discriminate between speakers. For example, loss functions may be applied on positive samples of speech utterances belonging to their associated speaker and negative samples of speech utterances not

associated with the speaker. For another example, center loss may be measured as the Euclidean loss between speaker embeddings and their centers.

[36] Embodiments described herein provide methods, systems, and apparatus for personalized speaker verification. In some embodiments, an online machine learning strategy for personalized speaker verification system is disclosed. Instead of fixing a universal speaker recognition model for all speakers, the online machine learning strategy may be used to train a personalized speaker recognition model for each target speaker, which helps to strengthen the robustness of speaker model and improve model performance. Online machine learning may refer to a method of machine learning in which data becomes available in a sequential order and is used to update the best predictor (model) for future data at each step, as opposed to batch learning techniques which generate the best predictor by learning on the entire training data set at once.

[37] FIG. 1 illustrates a system 100 for personalized speaker identification in accordance with some embodiments. The components of the system 100 presented below are intended to be illustrative. Depending on the implementation, the system 100 may include additional, fewer, or alternative components.

[38] In some embodiments, the system 100 may include a computing system 102, a computing device 104, a computing device 106, and a computing device 108. It is to be understood that although three computing devices are shown in FIG. 1, any number of computing devices may be included in the system 100. The computing system 102 may be implemented in one or more networks (e.g., enterprise networks), one or more endpoints, one or more servers (e.g., server 130), or one or more clouds. The server 130 may include hardware or software which manages access to a centralized resource or service in a network. A cloud may include a cluster of servers and other devices which are distributed across a network.

[39] In some embodiments, the computing system 102 may include a first obtaining component 112, a first feeding component 114, a second obtaining component 116, and a second feeding component 118, one or more of which may be optional. The computing system 102 may include other components. The various components of the computing system 102 may be integrated in one physical device or distributed in multiple physical devices.

[40] In some embodiments, the computing system 102 may include one or more processors (e.g., a digital processor, an analog processor, a digital circuit designed to process information, a central processing unit, a graphics processing unit, a microcontroller or microprocessor, an

analog circuit designed to process information, a state machine, and/or other mechanisms for electronically processing information) and one or more memories (e.g., permanent memory, temporary memory, non-transitory computer-readable storage medium). The one or more memories may be configured with instructions executable by the one or more processors. The processor(s) may be configured to perform various operations by interpreting machine-readable instructions stored in the memory. The computing system 102 may be installed with appropriate software (e.g., platform program, etc.) and/or hardware (e.g., wires, wireless connections, etc.) to access other devices of the system 100.

[41] In some embodiments, the computing devices 104, 106, and 108 may be implemented on or as various devices such as a mobile phone, tablet, server, desktop computer, and laptop computer. The computing device 106 and the computing device 108 may be the same device or different devices. The computing system 102 may communicate with the computing devices 104, 106, and 108, and other computing devices. Communication between devices may occur over the internet, through a local network (e.g., LAN), through direct communication (e.g., BLUETOOTH™, radio frequency, infrared), etc. In one embodiment, the computing device 104, 106, or 108 may comprise or be associated with a microphone or an alternative device configured to record speech data (e.g., human voices). A user may speak within the detection range of the microphone for the audio capture.

[42] In some embodiments, the system 100 may include a personalized speaker verification platform. For example, the computing system 102 and/or other computing devices may implement the personalized speaker verification platform. The personalized speaker verification platform may train a model for personalized speaker verification and effectuate its applications. For example, the platform may obtain training data from various sources, such as the computing device 104 (through communications 122) and/or the computing device 108 (through communications 128), to build a personalized speaker verification model for a target speaker 160.

[43] In some embodiments, the computing device 104 may have recorded or stored speech data of one or more speakers. For example, the computing device 104 may provide a speech data store. At a first stage, the platform may train a machine learning model with speech data obtained from the computing device 104 to obtain a first model (referred to as “a first model for determining voice characteristics”). The first model has not been personalized. At a second stage, the first model may be further trained with more training data (e.g., a positive sample and a

negative sample) to obtain a second model (referred to as “a second model for personalized speaker verification”).

[44] In some embodiments, the speaker 160 may interact with the computing device 108 to provide a portion of the training data (e.g., positive sample). For example, by speaking to a microphone coupled to the computing device 108, the user 160 may provide speech data (e.g., voice 142) to the computing device 108. The computing system 102 may obtain the speech data spoken by the speaker 160 from the computing device 108 as the positive sample. The computing system 102 may obtain speech data that is not spoken by the speaker 160 from the computing device 104 as the negative sample.

[45] In some embodiments, with the positive and negative samples, the first model may be further trained to obtain the second model for personalized speaker verification. The second model may be deployed in a remote server, cloud, client-side device, etc. For example, the second model may be deployed in the computing device 106, the computing device 108, or the computing system 102. In one embodiment, the computing device 106 may be installed with a software application, a web application, an Application Program Interface (API), or another suitable interface for invoking the second model. A user 140 may interact with the computing device 106, through which the second model is invoked. For example, by speaking to a microphone coupled to the computing device 106, the user 140 may provide speech data (e.g., voice 126) to the computing device 106, which inputs the speech data into the second model to verify if the user 140 is the speaker 160.

[46] While the computing system 102 is shown in FIG. 1 as a single entity, this is merely for ease of reference and is not meant to be limiting. One or more components or one or more functionalities of the computing system 102 described herein may be implemented in a single computing device or multiple computing devices. For example, the computing system 102 may incorporate the computing device 106, or vice versa. That is, each of the first obtaining component 112, first feeding component 114, second obtaining component 116, and second feeding component 118 may be implemented in the computing system 102 or the computing device 106 or 108. Similarly, the computing system 102 may couple to and associate with one or more other computing devices that effectuate a portion of the components or functions of the computing system 102. The computing device 106 may comprise one or more processors and

one or more memories coupled to the processors configured with instructions executable by one or more processors to cause the one or more processors to perform various steps described herein.

[47] The various components of the system 100 (e.g., the first obtaining component 112, the first feeding component 114, the second obtaining component 116, and the second feeding component 118 of the computing system 102) may be configured to perform steps for personalized speaker verification. FIGs. 2A-2B below describe the first model and the process of obtaining of a template vector through the first model. FIG. 2C below describes the process of obtaining the second model from the first model and applying the second model for personalized speaker verification.

[48] Referring to FIG. 2A, FIG. 2A illustrates a method for obtaining feature vectors in accordance with some embodiments. The method may be performed by one or more components of the system 100, such as the computing system 102 and/or the computing device 108. In some embodiments, the first obtaining component 112 may be configured to obtain speech data of the speaker 160, and the speech data may be processed and fed to the first model to output feature vectors of the speaker 160. In one embodiment, to obtain the speech data of the speaker 160, the first obtaining component 112 may be configured to obtain a spectrogram corresponding to the speech data, and obtain a plurality of feature vectors corresponding to the spectrogram as described below with reference to numerals 201-205.

[49] In some embodiments, as shown in FIG. 2A, audio queue 201 may represent an example of speech data of a speaker (e.g., the speaker 160) obtained from the computing device 108 or another device. The audio queue 201 is labelled with corresponding blocks of speech words, pauses (pau), or silences (sil) in a continuous time series in the x-axis direction. Vertical dash lines in the figure may mark the same timestamps on various illustrations and indicate the corresponding relationship among them. Depending on the application, the audio queue 201 may or may not be required to contain certain text-dependent trigger phrases.

[50] In some embodiments, audio queue 202 is an alternative representation of the audio queue 201, by breaking down the words into language units. There may be many classifications and definitions of language units, such as phonemes, phoneme portions, triphone, word, and n-gram. The language units shown are merely examples. In one example, phonemes are groups of speech sounds that have a unique meaning or function in a language, and may be the smallest meaningful contrastive unit in the phonology of a language. The number of phonemes may vary

for different languages, with most languages having 20-40 phonemes. In one example, “hello” can be separated into language units/phonemes “hh,” “ah,” “l,” and “ow.”

[51] In some embodiments, spectrum 203 may represent the speech data (e.g., the audio queue 201). There may be various different representations of the audio. In one example, the spectrum 203 may show the amplitude of captured sound with respect to time.

[52] In some embodiments, spectrogram 204 may be obtained based at least on the spectrum 203. The spectrogram 204 may be a frequency versus time representation of the speech data. In one embodiment, a Fourier transform may be applied to the spectrum 203 to obtain the spectrogram 204. In the spectrogram 204, the amplitude information is displayed in a greyscale as dark and bright regions. Bright regions may indicate that no sound was captured (e.g., pause, silence) at the corresponding time at the corresponding frequency, and dark regions may indicate the presence of sound. Based on the variation of the dark and bright patterns in the x-axis direction, boundaries between language units (e.g., words, phones) may be determined. Further, the pattern of dark regions in the y-axis direction between two dash lines may indicate the various frequencies captured at the corresponding time period and can provide information of the formants (carrying the identity of the sound) and transitions to help determine the corresponding phones.

[53] In some embodiments, a feature sequence 205 may be obtained based at least on the spectrogram 204. In one embodiment, cepstral analysis may be applied to the spectrogram 204 to obtain the feature sequence 205. For example, a time frame may move along the x-axis and sample the spectrogram 204 frame by frame. The speech data may thus be converted to a series of feature vectors shown as rectangular blocks in the figure. In one embodiment, short time spectrograms may be generated within a sliding Hamming window with width 25ms, step size 10ms and 1024-point FFT (fast Fourier transform). Mean and variance normalization may be performed along the frequency axis. 300-frame crop of audio speech data may be randomly sampled from each utterance for training that cover 3.015 seconds of speech and give spectrograms of size 300×512 , i.e., 300-dimension temporal and 512-dimension frequency feature. A person skilled in the art would appreciate the application of other techniques to obtain the feature vectors. These feature vectors may be used to train a model or be passed to a trained model for implementing various applications. For example, the feature vectors of the speaker 160 may be passed to a first model to output the corresponding voice characteristics. Before that,

the first model for determining voice characteristics and its training process is described below with reference to FIG. 2B.

[54] Referring to FIG. 2B, FIG. 2B illustrates an example of a method for training a model for determining voice characteristics (the first model) in accordance with some embodiments. The method may be performed by one or more components of the system 100, such as the computing system 102 and/or the computing device 104. The first model described herein such as FIG. 2B is merely an example and is not intended to limit the structure or training process. A person of ordinary skill in the art would appreciate the use of other speaker embedding models for the first model or in conjunction with the systems and method disclosed herein.

[55] In some embodiments, as shown, the first model may be a deep learning model comprising a plurality of layers. As a subset of machine learning, deep learning may utilize a hierarchical level of artificial neural networks to carry out the process of machine learning. The hierarchical function of deep learning systems enables machines to process data with a nonlinear approach. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. An artificial neural network is based on a collection of connected units or nodes called artificial neurons (represented by circles in various layers such as layers 207 and 208 in this figure), which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal other neurons connected to it. The signal at a connection may be a real number, and the output of each neuron may be computed by some non-linear function of the sum of its inputs. The connections are called edges (represented by connecting lines such as those between layers 207 and 208 in this figure). Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Neurons are aggregated into layers. Since each layer comprises a plurality of neurons, neighboring layers are connected by various neuron-to-neuron connections with associated weights. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers one or more times.

[56] In some embodiments, as an overview of FIG. 2B, a DNN can be used as feature extractor for taking the cepstral acoustic features (e.g., feature sequence 205) as its input, uses several

layers of frame-level forward or convolution operations, and then after a pooling layer, outputs a segment-level representation known as embedding. A combination of classification and clustering loss is used to train embeddings. With the embedding, a softmax classifier with a projection from embedding to speaker IDs may be used to distinguish different speakers. Also, an annealing scheme uses margin in the classification loss to improve generalization capability of the trained embedding and make the training process more stable. Since different speakers may result in different Gaussians distributions with different means and standard deviations, reflecting the distinctiveness of human voices, the trained embeddings may be distributed in mixture of Gaussians with multiple shapes and modes. To drive the trained embeddings towards this distribution, the clustering loss is applied. Further details of the first model are described below.

[57] In some embodiments, the trained or untrained first model may include a plurality of neuron layers outputting from one to a next, forming the DNN. The plurality of neuron layers may comprise, for example, a ResNet-34 (34-layer residual network) architecture, ResNet-50 architecture, etc. For example, the plurality of layers may include: a first convolution layer 206a configured to receive the plurality of feature vectors (e.g., the feature sequence 205) as an input of the first convolution layer 206a; a first pooling layer 206b configured to receive an output of the first convolution layer 206a as an input of the first pooling layer 206b; a plurality of residual network layers 206c configured to receive an output of the first pooling layer 206b as an input of the plurality of residual network layers 206c; a second convolution layer 206d configured to receive an output of the plurality of residual network layers 206c as an input of the second convolution layer 206d; a second pooling layer 207 configured to receive an output of the second convolution layer 206d as an input of the second pooling layer 207; and an embedding layer 208 configured to receive an output of the second pooling layer 207 as an input of the embedding layer 208 and output a vector representing the one or more voice characteristics of the speaker. The first convolution layer 206a may be the input layer, and the embedding layer 208 may be the output layer. The first convolution layer 206a, the first pooling layer 206b, the plurality of residual network layers 206c, and the second convolution layer 206d may be referred to a shared network 206.

[58] An embedding is a mapping of a discrete variable to a vector of continuous numbers. In some embodiments, through the embedding layer 208, words or phrases of the speech data input

may be mapped to vectors of real numbers. Thus, the first model transforms from a space with many dimensions per word to a continuous vector space with a much lower dimension.

[59] In some embodiments, a ResNet-34 (34-layer residual network) architecture as shown in Table 1 may be used. In Table 1, conv1 may correspond to the first convolution layer 206a, pool1 may correspond to the first pooling layer 206b, res_block1 to res_block 4 may correspond to the plurality of residual network layers 206c, conv2 may correspond to the second convolution layer 206d, and pool2 may correspond to the second pooling layer 207. For the output of every convolutional operator, batch normalization, not shown in Table 1, is applied before computing rectified linear unit (ReLU) activations. The parameters of ResNet-34 may be initialized. The embedding size may be set to 512, and the 512-channel parametric ReLU (PReLU) activations may be used as feature embedding. Using PReLU as nonlinear activation function has advantages of avoiding canceling correlation in negative dimensions of embedding space like ReLU and strengthening the robustness of embedding feature.

Layer Name	Configuration	Output Size
conv1	$(7 \times 7, 64)$, stride 2	$64 \times 148 \times 254$
pool1	3×3 , max pool, stride 2	$64 \times 73 \times 126$
res_block1	$[(3 \times 3, 64)_2] \times 3$	$64 \times 73 \times 126$
res_block2	$[(3 \times 3, 128)_2] \times 4$	$128 \times 37 \times 63$
res_block3	$[(3 \times 3, 256)_2] \times 6$	$256 \times 19 \times 32$
res_block4	$[(3 \times 3, 512)_2] \times 3$	$512 \times 10 \times 16$
conv2	$(1 \times 9, 512)$	$512 \times 10 \times 8$
pool2	adaptive average pool	$512 \times 1 \times 1$
embedding	512×512 , PReLU	$512 \rightarrow 512$
classification	$512 \times C$, $C = 5994$	$512 \rightarrow 5994$

Table 1. The ResNet-34 architecture. The triple output size is in the form of (*channel* $\times$ *temporal* $\times$ *frequency*). $[(3 \times 3, 64)_2] \times 3$ means 3 residual blocks, each of which comprising 2 convolutional operators, each with kernel size 3×3 and 64 filters, others in analogy. For the first block of res_block2 ~ 4 with different numbers of filters between the input and output, a short cut connection is needed, using one convolution with kernel size 1×1 .

[60] In some embodiments, if the first model is untrained, the first model can be trained at least by jointly minimizing a first loss function and a second loss function. For example, the feature sequence 205 may be fed to the untrained first model to train the layers (e.g., from the input layer to the output layer) by minimizing the two loss functions. Minimizing loss function a method of evaluating how well specific algorithm models the given data. If predictions deviate too much from actual results, loss function would cough up a very large number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction.

[61] The first loss function (209a for classification) is introduced below. In some embodiments, the first loss function (e.g., Equation (3) below) may be a non-sampling-based loss function. Training the first model by minimizing the first loss function optimizes speaker classification. The loss function for classification may be a computationally feasible loss function representing the price paid for inaccuracy of predictions in identifying which category (e.g., speaker identity category) a particular observation (e.g., speech data input) belongs to. The goal of the learning process, often involving labeled datasets, is to minimize expected risk.

[62] In one embodiment, the non-sampling-based loss function comprises an additive margin softmax loss function. A softmax function takes an N-dimensional vector of real numbers and transforms it into a vector of real number in range (0, 1) which add up to 1. The softmax function may be used in the final layer of a neural network-based classifier. Such networks may be trained under a log loss or cross-entropy regime, giving a non-linear variant of multinomial logistic regression. Compared to the original softmax which separates two different classes with a decision boundary line, additive margin softmax separates two different classes with a decision margin (e.g., an angular area).

[63] In some embodiments, minimizing the first loss function comprises, for at least the embedding layer, minimizing a non-sampling-based loss function to optimize between-class classification error. That is, the error of mixing up one class with another is minimized. By the classification error optimization, classes are made further apart (e.g., class spk1 (speaker 1) and class spk3 (speaker 3) of 209a are far apart) and easier to distinguish from one another, thereby reducing the chance for mix-up in applications. In one embodiment, minimizing the first loss function trains the plurality of neuron layers (e.g., from the first convolution layer 206a to the embedding layer 208). This provides one end-to-end framework to train the first model, instead of training the first convolution layer 206a to the embedding layer 208 under one model, and training the embedding layer 208 for optimizing classification or clustering under another model.

[64] A softmax loss function for classification is formulated as follows

$$\mathcal{L}_{Softmax} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\mathbf{w}_y^T \mathbf{x}_i}}{\sum_{j=0}^{C-1} e^{\mathbf{w}_j^T \mathbf{x}_i}}, \quad (1)$$

where N and C are the batch size and the class number respectively. $\mathbf{x}_i \in \mathbb{R}^{d+1}$ is a $d+1$ dimensional real vector with d dimension embedding of the i^{th} sample, appended with a scalar 1.

$w_j \in \mathbb{R}^{d+1}$ is the weight vector for class j . The inner product $w_j^T x_i$ can be equivalently expressed in angular form as $\|w_j\| \|x_i\| \cos(\theta_{x_i, w_j})$, where θ_{x_i, w_j} is the angle between w_j and x_i .

[65] Angular softmax introduces a multiplicative margin m on the angle θ . Angular softmax first constrains the weight vector w_j to have unit-norm, i.e., normalize the weights and zero the bias scalar in w_j ($\|w_{j,i < d}\| = 1, w_{j,d} = 0$). The inner product becomes $\|x_i\| \cos(\theta_{x_i, w_j})$. It further applies the multiplicative margin as follows

$$\begin{aligned} \mathcal{L}_{AS} &= -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\|x_i\| \cos(m\theta_{x_i, w_{y_i}})}}{Z_{x_i}}, \\ Z_{x_i} &= e^{\|x_i\| \cos(m\theta_{x_i, w_{y_i}})} + \sum_{j=0, j \neq y_i}^{C-1} e^{\|x_i\| \cos(\theta_{x_i, w_j})}. \end{aligned} \quad (2)$$

[66] Here, m is applied to the positive label y_i only. For angle $\theta_{x_i, w_{y_i}}$ between x_i and its corresponding label y_i in $(-\frac{\pi}{2m}, \frac{\pi}{2m})$, choosing m larger than 1.0 reduces the angular distance $\cos(m\theta_{x_i, w_{y_i}})$. When annealing m gradually from 1.0 to a larger value during training, it can force the learned embedding x_i for corresponding label y_i to be more discriminative than trained from softmax.

[67] Instead of using the multiplicative margin m as in Equation (2), additive margin softmax uses an additive margin in angular space. Furthermore, embedding x_i is normalized to be one ($\|x_i\| = 1$) and then rescaled by a hyperparameter s . The loss function is as follows

$$\begin{aligned} \mathcal{L}_{AMS} &= -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s\psi(\theta_{x_i, w_{y_i}})}}{Z_{x_i}}, \\ Z_{x_i} &= e^{s\psi(\theta_{x_i, w_{y_i}})} + \sum_{j=0, j \neq y_i}^{C-1} e^{s \cos(\theta_{x_i, w_j})}, \end{aligned} \quad (3)$$

[68] $\psi(\theta)$ has two forms of definitions. One is defined as $\cos \theta - m$ for additive cosine margin softmax, i.e., *CosAMS*. The second is $\cos(\theta + m)$ for additive angle margin softmax, i.e., *ArcAMS*. Increasing m would result in reduced posterior in Equation (3) as cosine function is monotonically decreasing, therefore forcing x_i to be more discriminative. Additionally, s can be considered as a temperature parameter for annealing. Using a large s makes the posterior sharper than using $s = 1$. In some embodiments, the first loss function includes the additive margin softmax loss function $\mathcal{L}_{AMS}$ (Equation 3).

[69] Training models with discriminative loss function such as large margin one may suffer from local optimum or divergence. A current approach to handle this is starting from a pre-trained first model with softmax loss, but this can be time-consuming. In some embodiments, the annealing method for training models with additive margin softmax loss is introduced below. In one embodiment, minimizing the first loss function comprises increasing a margin linearly from zero to a target margin value for annealing. The training processes stabilize as one progressive transition for the margin m . The margin m may be increased linearly from 0 to the target margin value as

$$m = \min(m_{max}, m_{inc} \times t),$$

where $t \geq 0$ is the epoch step. To guarantee numerical stability for the *ArcAMS* loss, it may be configured to $\psi(\theta) = \cos(\theta + m)$ if $\sin(\theta + m) > 0$ (that is, in the upper quadrant of Cartesian coordinated system) or $\psi(\theta) = \cos(\theta)$ if otherwise.

[70] The second loss function (209b for clustering) is introduced below. In some embodiments, the second function may be a Gaussian mixture loss function with non-unit multi-variant covariance matrix. In one embodiment, the non-unit multi-variant covariance matrix comprises a standard deviation diagonal matrix. A covariance matrix is a matrix whose element in the i, j position is the covariance between the i^{th} and j^{th} elements of a random vector.

[71] In some embodiments, training the first model by minimizing the second loss function optimizes speaker clustering. The goal of clustering is to group similar data points together without supervision or prior knowledge of nature of the clusters. In some embodiments, the loss function for clustering may be a linear combination of unsupervised representation learning loss and a clustering oriented loss. See Equations (6) to (8) for more details.

[72] In some embodiments, minimizing the second loss function comprises, for at least the embedding layer, minimizing a Gaussian mixture loss function with non-unit multi-variant covariance matrix to reduce intra-class variation. For Gaussian mixture loss function, each cluster is modelled according to a different Gaussian distribution. Each data point may be generated by any of the distributions with a corresponding probability. By the clustering variation loss optimization, units in the same classes more strongly resemble each other (e.g., class spk1 of 209b is small in size and reflects strong resemblance among its units). In one

embodiment, the Gaussian mixture loss function with non-unit multi-variant covariance matrix comprises a large margin Gaussian mixture loss function.

[73] In one embodiment, minimizing the second loss function trains the plurality of neuron layers (e.g., from the first convolution layer 206a to the embedding layer 208). This provides one end-to-end framework to train the first model, instead of training the first convolution layer 206a to the embedding layer 208 under one model, and training the embedding layer 208 for optimizing classification or clustering under another model.

[74] In some embodiments, it is assumed that the extracted embedding x_i on the training set is distributed as mixtures of Gaussian densities. Each Gaussian component k has its mean μ_k and covariance Σ_k with prior probability π_k . If there are C such Gaussian components, a loss $\mathcal{L}_{cls}$ is defined to measure the closeness of the hypothesis of x_i belonging to cluster k and the posterior probability from the Gaussian mixture model. The posterior probability is denoted as $\gamma_k(y_i) = p(y_i = k|x_i) = \frac{N(x_i; \mu_k, \Sigma_k) \pi_k}{\sum_{k=0}^{C-1} N(x_i; \mu_k, \Sigma_k) \pi_k}$. Aggregating this over all observations gives:

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{i=1}^N \log \gamma_i(y_i = k). \quad (4)$$

Speaker ID of x_i for y_i and C corresponds to the class number as classification task. Then, in the embedding space, $\mathcal{L}_{cls}$ focuses more on discriminative capability.

[75] However, optimizing the above loss cannot ensure that the obtained embedding x_i fits a mixture of Gaussian distribution. Thus, in some embodiments, a regularization term that explicitly drives the embedding towards a mixture of Gaussian density distribution may be added by introducing a likelihood function as follows

$$\mathcal{L}_{likelihood} = -\frac{1}{N} \sum_{i=1}^N \log (\mathcal{N}(x_i; \mu_{y_i}, \Sigma_{y_i}) \pi_{y_i}). \quad (5)$$

Increasing this likelihood can drive the extracted embedding x_i towards its corresponding Gaussian distribution.

[76] In some embodiments, the Gaussian mixture loss $\mathcal{L}_{GM}$ may be defined as

$$\mathcal{L}_{GM} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{likelihood}, \quad (6)$$

in which λ is a non-negative weighting coefficient. For simplicity, the prior $\pi_k = \frac{1}{C}$ and Σ_k may be diagonal covariance matrix. $\mathcal{L}_{GM}$ then becomes the following in which constant terms are removed.

$$\begin{aligned} \mathcal{L}_{GM} = & \frac{1}{N} \sum_{i=1}^N \left(-\log \frac{|\Sigma_{y_i}|^{-\frac{1}{2}} e^{-d_{x_i, y_i}}}{\sum_{k=0}^{C-1} |\Sigma_k|^{-\frac{1}{2}} e^{-d_{x_i, k}}} + \right. \\ & \left. \lambda(d_{x_i, y_i} + \frac{1}{2} \log |\Sigma_{y_i}|) \right), \quad (7) \\ d_{x_i, k} = & \frac{1}{2} (\mathbf{x}_i - \mu_k)^T \Sigma_k^{-1} (\mathbf{x}_i - \mu_k), \end{aligned}$$

[77] In some embodiments, to optimize loss $\mathcal{L}_{GM}$, all of the parameters for Gaussian mixture components, including μ_k and Σ_k , and embedding x_i are updated using stochastic gradient descent (SGD) algorithm. Applying diagonal covariance in Equation (7) may have numerical difficulty, because the covariance matrix Σ_k needs to be positive semi-definite. In some embodiments, it is defined that $\Sigma_k = \Lambda_k^2$. Instead of Σ_k , the standard deviation diagonal matrix Λ_k is the parameter to learn. Λ_k is initialized to be identity matrix.

[78] In some embodiments, when Gaussian component identities are given, it is beneficial to apply margin to improve generalization capability of learned embedding x_i . To this end, the distance d_{x_i, y_i} is increased for positive samples with a scalar $1 + \alpha_i$, with margin α being larger than zero. The new loss function, large margin Gaussian mixture, is defined as follows

$$\begin{aligned} \mathcal{L}_{LMGM} = & \frac{1}{N} \sum_{i=1}^N \left(-\log \frac{|\Sigma_{y_i}|^{-\frac{1}{2}} e^{-d_{x_i, y_i} (1 + \alpha)}}{Z_{x_i}} + \right. \\ & \left. \lambda(d_{x_i, y_i} + \frac{1}{2} \log |\Sigma_{y_i}|) \right), \quad (8) \\ Z_{x_i} = & \sum_{k=0}^{C-1} |\Sigma_k|^{-\frac{1}{2}} e^{-d_{x_i, k} (1 + \alpha I_{k=y_i})}, \end{aligned}$$

where I_{cond} is an indicator function equal to 1 if *cond* is satisfied or 0 if otherwise. In some embodiments, the second loss function includes the Gaussian mixture loss function with non-unit multi-variant covariance matrix $\mathcal{L}_{LMGM}$ (Equation 8).

[79] In some embodiments, the first loss function acts as a regularizer to the second loss function, and the second loss function acts as a regularizer to the first loss function. Regularizers allow to apply penalties on layer parameters or layer activity during optimization. These

penalties are incorporated in the loss function that the network optimizes. These two loss functions help each other to achieve stable and fast convergence when training embeddings.

[80] In some embodiments, the integral loss function is a combination of classification and clustering loss, represented as the addition of Equation (3) and Equation (8), such as follows

$$Loss = \mathcal{L}_{AMS} + \mathcal{L}_{LMGM}. \quad (9)$$

Equation (9) may use $m_{max} = 0.2$, $m_{inc} = 0.035$, $s = 30$ for $\mathcal{L}_{CosAMS}$ loss and $m_{max} = 0.25$, $m_{inc} = 0.045$, $s = 30$ for $\mathcal{L}_{ArcAMS}$ loss. For $\mathcal{L}_{LMGM}$ loss, let $\alpha = 0.01$ and $\lambda = 0.01$.

[81] In some embodiments, two metrics may be used for model evaluation. The first metrics is Equal Error Rate (EER), defined as the rate at which miss probability P_{miss} is equal to false alarm probability P_{fa} , and the second is Minimum Detection Cost Function (minDCF), defined as $C_{det}^{min} = \min(C_{miss} \times P_{miss} \times P_{tar} + C_{fa} \times P_{fa} \times (1 - P_{tar}))$, where C_{miss} and C_{fa} are the cost weights of P_{miss} and P_{fa} respectively, and P_{tar} is the target probability. Since P_{miss} and P_{fa} are functions of threshold, the minimum in C_{det}^{min} is computed over all possible thresholds. For example, $C_{miss} = 1.0$, $C_{fa} = 1.0$, and $P_{tar} = 0.01$.

[82] In some embodiments, for training and test, models are optimized with momentum SGD, which has a momentum of 0.9 and weight decay of 5×10^{-4} . Mini-batch size is 64. At the beginning of each epoch, training samples are randomly shuffled. The initial learning rate is 0.1 for additive margin softmax loss, including $\mathcal{L}_{CosAMS}$ and $\mathcal{L}_{ArcAMS}$ ones. For additive margin softmax loss and $\mathcal{L}_{LMGM}$, the learning rate is set to 0.01. Learning rate is decayed by 0.1 every 6 epoch. After 12 epochs, training is terminated to avoid overfitting when the Equal Error Rate (EER) is increasing on the validation set. During test, utterances with whole length are used, and the embeddings are extracted with adaptive average pooling in Table 1. As end-to-end deep speaker embedding model, cosine distance is used as backend scoring when evaluating performance.

[83] As described, the first model may be trained based at least on optimizing two jointly employed loss functions. In some embodiments, a non-sampling-based loss function is employed for optimizing between-class separability, and a Gaussian mixture loss with non-unit multi-variant covariance matrix is employed for reducing intra-class variation. This combination not only improves the generalization capability of classification using margin-based methods, but also explicitly reduces intra-class variability. In one embodiment, optimization for both between-

class separability and intra-class variability achieves better performance (e.g., faster and better convergence) than optimization for class separability or intra-class variability alone. In one embodiment, an end-to-end framework is provided to train the first model through minimizing the two loss functions. In some embodiments, with the joint optimization, the learned embeddings capture segment-level acoustic representation from variable-length speech segments to discriminate between speakers and to replicate densities of speaker clusters. In one embodiment, greater similarities for subjects in the same class and greater differences for subjects in different classes may be realized, making the determined voice characteristics more accurate. In one embodiment, the optimization for intra-class variability improves the process of voice characteristics determination by accommodating the possibility that speaker embeddings are distributed with multiple shapes and modes. In one embodiment, to optimize for between-class separability, an annealing method is provided to train additive margin softmax loss stably.

[84] Referring to FIG. 2C, FIG. 2C illustrates a method 290 for personalized speaker verification in accordance with some embodiments. The method 290 may be performed by one or more components of the system 100, such as the computing system 102. The method 290 may implement an online machine learning strategy. The steps of the method 290 presented below are intended to be illustrative. Depending on the implementation, the method may include additional, fewer, or alternative components.

[85] In some embodiments, the first obtaining component 112 may be configured to obtain first speech data of a speaker as a positive sample 241 and second speech data of an entity different from the speaker as a negative sample 242. For example, to personalize speaker verification for the speaker 160, the first speech data may be spoken by the speaker 160, and the second speech data may be spoken by an entity other than the speaker 160. The second speech data may be spoken by a different person or by a machine. The second speech data may be, for example, chosen at random from a corpora database. The speech data may be in any data format conveying speech information of the speaker, such as a speech record in a digital file.

[86] In some embodiments, at step 221, the first feeding component 114 may be configured to feed the positive sample and the negative sample to the first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker. In one embodiment, the first feeding component 114 may be configured to input the positive sample 241 and the negative sample 242, together or separately,

into the first model 243. As described above with reference to FIG. 2A, the first feeding component 114 may be configured to convert the speech data into a plurality of corresponding feature vectors to input into the first model.

[87] As described above with reference to FIG. 2B, the first model may have been trained at least by jointly minimizing the first loss function and the second loss function. The first model may include the layers 206a to 208, but with the associated parameters such as the weights trained. The first model may have been optimized for classification and clustering, but not personalized for speaker verification. Thus, although the associated parameters such as the weights have been trained for the first model, the associated parameters may be further trained as described below when updating the first model to obtain the second model for personalized speaker verification.

[88] In some embodiments, at step 222, the first model 243 may output one or more voice characteristics. The one or more voice characteristics may be represented by the embedding layer output from the first model. Based on the positive sample 241 of the speaker 160, the first model 243 may output a positive voice characteristic 244. Based on the negative sample 242, the first model 243 may output a negative voice characteristic 245. The generation of the positive and negative voice characteristics may be independent of each other and may be performed separately.

[89] In some embodiments, the positive and negative speech data may each comprises one or more speech segments, such as one or more phrases uttered by a speaker. That is, the first obtaining component 112 may be configured to obtain one or more speech segments (e.g., P1, P2 ..., Pk) of the speaker as the positive sample and one or more speech segments (e.g., N1, N2 ..., Nk) of one or more people other than the speaker as the negative sample. Accordingly, the first feeding component 114 may be configured to feed the one or more speech segments of the speaker into the first model to correspondingly output one or more positive sample vectors (e.g., PV1, PV2, ..., PVk) from the embedding layer 208, and feed the one or more speech segments of the one or more people other than the speaker into the first model to correspondingly output one or more negative sample vectors (e.g., NV1, NV2, ..., NVk) from the embedding layer 208. For example, P1 may be input to the first model to output PV1, and the process may be repeated for all the k speech segments. The same may be applied to the negative sample.

[90] In some embodiments, at step 223, the computing system 102 may be configured to merge the one or more positive sample vectors 244. For example, the computing system 102 may be configured to average the positive sample vectors (PV1, PV2, ..., PVk) to obtain a template vector 246 (PVX) of the speaker. The template vector may also be referred to as a voiceprint template vector of the speaker 160.

[91] In some embodiments, at steps 224 and 225, the second obtaining component 116 may be configured to obtain a gradient 248 based at least on the positive voice characteristic 244 and the negative voice characteristic 245. To this end, at step 224, the second obtaining component 116 may be configured to feed the one or more positive sample vectors (e.g., PV1, PV2, ..., PVk) and the one or more negative sample vectors (e.g., NV1, NV2, ..., NVk) into a neural network classifier to obtain one or more gradient vectors 247 (e.g., G1, G2, ..., Gk). For example, with inputs PV1 and NV1, the neural network classifier may output G1, and the process may be repeated for all k vectors. A neural network classifier may include of units (neurons) arranged in layers, which produces class labels as output from a set of features of an object.

[92] In some embodiments, the positive and negative sample vectors may have a predetermined proportion (e.g., 1:5, 1:10, etc.). For example, at step 221, for a first sample set, the first feeding component 114 may be configured to feed one phrase uttered by the speaker 160 (as a part of the positive sample) and five phrases uttered by someone other than the speaker 160 (as a part of the negative sample) to the first model to correspondingly output, at step 222, one positive sample vector PV1 (as a part of the positive voice characteristic) and five negative sample vectors NV1a – NV1d (as a part of the negative voice characteristic). Then, at step 224, the second obtaining component 116 may be configured to feed the one positive sample vector PV1 and the five negative sample vectors NV1a – NV1d into the neural network classifier to obtain one gradient vector G1. The same process may be repeated for k sample sets to correspondingly obtain the gradient vectors 247 (e.g., G1, G2, ..., Gk).

[93] In some embodiments, the second feeding component 118 may be configured to obtain the gradient based at least on backward propagation through a cross-entropy loss function of the neural network classifier. The neural network classifier may be a binary classifier that labels each positive sample vector as 1 and labels each negative sample vector as 0. The gradient of the cross-entropy loss function may include, for example, partial derivatives of the cross-entropy loss function with respect to each parameter.

[94] In some embodiments, at step 225, the second obtaining component 116 may be further configured to merge the one or more gradient vectors 247. For example, the second obtaining component 116 may be further configured to average the one or more gradient vectors (e.g., G_1 , G_2 , ..., G_k) to obtain an average gradient vector of the speaker (GX) as the gradient 248.

[95] In some embodiments, at step 226, the second feeding component 118 may be configured to feed the gradient 248 to the first model 243 to update one or more parameters of the first model 243 to obtain a second model 249 for personalized speaker verification. For example, the second feeding component 118 may be configured to feed the average gradient vector (GX) 248 of the speaker to the first model 243 to update the one or more parameters of the first model 243. The one or more parameters (e.g., the weights) may associate different neural layers of the first model 243.

[96] In some embodiments, the second feeding component 118 may be configured to feed the gradient 248 to the first model 243 to update the one or more parameters of the first model 243 based at least on the gradient 248 and an online machine learning rate. For example, the one or more parameters may be updated based on a stochastic gradient descent (SGD) algorithm. In one embodiment, the second feeding component 118 may be configured to update the one or more parameters in a direction in which the gradient descends at the fastest online machine learning rate. For example, to update a parameter p , p may be iteratively updated according to $p = (p - LR * GX)$, where LR is the online machine learning rate and GX is the average gradient vector. The online learning rate may indicate the descent step size. For example, the online learning rate may refer to the relative amount that the parameters are updated during online training.

[97] In some embodiments, at step 227, the computing system 102 may be configured to obtain speech data of a user 140, and feed the obtained speech data to the second model 249 to obtain an input vector 251 (IV) of the user. While the positive/negative sample vectors are output from the embedding layer 208 of the first model, the input vector (IV) may be output from the embedding layer of the second model.

[98] In some embodiments, at step 229, the computing system 102 may be configured to compare the input vector 251 (IV) of the user 140 with the template vector 246 (PVX) of the speaker 160, and verify if the user 140 is the speaker 160 based at least on the comparison. In one embodiment, the computing system 102 may be configured to compare, with a threshold, a distance between the input vector 251 (IV) of the user 140 with the template vector 246 (PVX)

of the speaker 160. If the distance is within the threshold, it indicates that the user 140 is likely the speaker 160. If the distance is not within the threshold, it indicates that the user 140 is unlikely speaker 160.

[99] In some embodiments, the speaker 160 may be an employee of a company who has enrolled in personalized speaker identification by providing personal speech data. The company's security gate may have installed or coupled to the computing system 102 implementing the first model and second model. Later, the speech data of a user speaking to the company's security gate may be used to verify if the user is the employee. If the employee's identity is verified, the employee's work time may be recorded accordingly.

[100] In some embodiments, the speaker 160 may have enrolled in personalized speaker identification with an application installed in her mobile phone. The mobile phone may have installed or coupled to the computing system 102 implementing the first model and second model. Later, the speech data of any user speaking to the mobile phone or another device implementing the application may be used to authenticate the user. If the mobile phone verifies that user is the speaker 160 (i.e., true owner) of the enrolled identity, the mobile phone may unlock certain functions for the user.

[101] FIG. 3 illustrates a flowchart of a method 300 for personalized speaker verification in accordance with some embodiments. The method 300 may be performed by a device, apparatus, or system for personalized speaker verification. The method 300 may be performed by one or more components of the environment or system illustrated by FIGs. 1-2C, such as the computing system 102. The operations of the method 300 presented below are intended to be illustrative. Depending on the implementation, the method 300 may include additional, fewer, or alternative steps performed in various orders or in parallel.

[102] Block 310 includes obtaining first speech data of a speaker as a positive sample and second speech data of an entity different from the speaker as a negative sample.

[103] Block 320 includes feeding the positive sample and the negative sample to a first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker.

[104] Block 330 includes obtaining a gradient based at least on the positive voice characteristic and the negative voice characteristic.

[105] Block 340 includes feeding the gradient to the first model to update one or more parameters of the first model to obtain a second model for personalized speaker verification.

[106] In some embodiments, obtaining the first speech data of the speaker as the positive sample and the second speech data of the entity different from the speaker as the negative sample comprises: obtaining one or more speech segments of the speaker as the positive sample and obtaining one or more speech segments of one or more people other than the speaker as the negative sample. Feeding the positive sample and the negative sample to the first model to correspondingly output the positive voice characteristic and the negative voice characteristic of the speaker comprises: feeding the one or more speech segments of the speaker into the first model to correspondingly output one or more positive sample vectors, and feeding the one or more speech segments of the one or more people other than the speaker into the first model to correspondingly output one or more negative sample vectors.

[107] In some embodiments, the method further comprises averaging the one or more positive sample vectors to obtain a template vector of the speaker.

[108] In some embodiments, the method further comprises: obtaining speech data of a user; feeding the obtained speech data to the second model to obtain an input vector of the user; comparing the input vector of the user with the template vector of the speaker; and verifying if the user is the speaker based at least on the comparison.

[109] In some embodiments, obtaining the gradient based at least on the positive voice characteristic and the negative voice characteristic comprises: feeding the one or more positive sample vectors and the one or more negative sample vectors into a neural network classifier to obtain one or more gradient vectors.

[110] In some embodiments, obtaining the gradient based at least on the positive voice characteristic and the negative voice characteristic further comprises: averaging the one or more gradient vectors to obtain an average gradient vector of the speaker as the gradient.

[111] In some embodiments, feeding the gradient to the first model to update the one or more parameters of the first model comprises: feeding the average gradient vector of the speaker to the first model to update the one or more parameters of the first model; and the one or more parameters associate different neural layers of the first model.

[112] In some embodiments, feeding the one or more positive sample vectors and the one or more negative sample vectors into the neural network classifier to obtain one or more gradient

vectors comprises: obtaining the gradient based at least on backward propagation through a cross-entropy loss function of the neural network classifier.

[113] In some embodiments, feeding the gradient to the first model to update the one or more parameters of the first model comprises: feeding the gradient to the first model to update the one or more parameters of the first model based at least on the gradient and an online machine learning rate.

[114] In some embodiments, feeding the gradient to the first model to update the one or more parameters of the first model based at least on the gradient and the online machine learning rate comprises: updating the one or more parameters in a direction in which the gradient descends at a fastest online machine learning rate.

[115] In some embodiments, before feeding the positive sample and the negative sample to the first model for determining voice characteristics, the first model has been trained at least by jointly minimizing a first loss function that optimizes speaker classification and a second loss function that optimizes speaker clustering.

[116] In some embodiments, the first loss function comprises a non-sampling-based loss function; and the second function comprises a Gaussian mixture loss function with non-unit multi-variant covariance matrix.

[117] FIG. 4 illustrates a block diagram of a computer system 400 apparatus for personalized speaker verification in accordance with some embodiments. The components of the computer system 400 presented below are intended to be illustrative. Depending on the implementation, the computer system 400 may include additional, fewer, or alternative components.

[118] The computer system 400 may be an example of an implementation of one or more components of the computing system 102. The method 300 may be implemented by the computer system 400. The computer system 400 may comprise one or more processors and one or more non-transitory computer-readable storage media (e.g., one or more memories) coupled to the one or more processors and configured with instructions executable by the one or more processors to cause the system or device (e.g., the processor) to perform the above-described method, e.g., the method 300. The computer system 400 may comprise various units/modules corresponding to the instructions (e.g., software instructions).

[119] In some embodiments, the computer system 400 may be referred to as an apparatus for personalized speaker verification. The apparatus may comprise a first obtaining module 410 for

obtaining first speech data of a speaker as a positive sample and second speech data of an entity different from the speaker as a negative sample; a first feeding module 420 for feeding the positive sample and the negative sample to a first model for determining voice characteristics to correspondingly output a positive voice characteristic and a negative voice characteristic of the speaker; a second obtaining module 430 for obtaining a gradient based at least on the positive voice characteristic and the negative voice characteristic; and a second feeding module 440 for feeding the gradient to the first model to update one or more parameters of the first model to obtain a second model for personalized speaker verification. The first obtaining module 410 may correspond to the first obtaining component 112. The first feeding module 420 may correspond to the first feeding component 114. The second obtaining module 430 may correspond to the second obtaining component 116. The second feeding module 440 may correspond to the second feeding component 118.

[120] The techniques described herein may be implemented by one or more special-purpose computing devices. The special-purpose computing devices may be desktop computer systems, server computer systems, portable computer systems, handheld devices, networking devices or any other device or combination of devices that incorporate hard-wired and/or program logic to implement the techniques. The special-purpose computing devices may be implemented as personal computers, laptops, cellular phones, camera phones, smart phones, personal digital assistants, media players, navigation devices, email devices, game consoles, tablet computers, wearable devices, or a combination thereof. Computing device(s) may be generally controlled and coordinated by operating system software. Conventional operating systems control and schedule computer processes for execution, perform memory management, provide file system, networking, I/O services, and provide a user interface functionality, such as a graphical user interface (“GUI”), among other things. The various systems, apparatuses, storage media, modules, and units described herein may be implemented in the special-purpose computing devices, or one or more computing chips of the one or more special-purpose computing devices. In some embodiments, the instructions described herein may be implemented in a virtual machine on the special-purpose computing device. When executed, the instructions may cause the special-purpose computing device to perform various methods described herein. The virtual machine may include a software, hardware, or a combination thereof.

[121] FIG. 5 illustrates a block diagram of a computer system 500 in which any of the embodiments described herein may be implemented. The computer system 500 may be implemented in any of the components of the devices, apparatuses, or systems illustrated in FIGs. 1-4, such as the computing system 102. One or more of the methods illustrated by FIGs. 1-4, such as the method 300, may be performed by one or more implementations of the computer system 500.

[122] The computer system 500 may include a bus 502 or other communication mechanism for communicating information, one or more hardware processor(s) 504 coupled with bus 502 for processing information. Hardware processor(s) 504 may be, for example, one or more general purpose microprocessors.

[123] The computer system 500 may also include a main memory 506, such as a random-access memory (RAM), cache and/or other dynamic storage devices, coupled to bus 502 for storing information and instructions executable by processor(s) 504. Main memory 506 also may be used for storing temporary variables or other intermediate information during execution of instructions executable by processor(s) 504. Such instructions, when stored in storage media accessible to processor(s) 504, render computer system 500 into a special-purpose machine that is customized to perform the operations specified in the instructions. The computer system 500 may further include a read only memory (ROM) 508 or other static storage device coupled to bus 502 for storing static information and instructions for processor(s) 504. A storage device 510, such as a magnetic disk, optical disk, or USB thumb drive (Flash drive), etc., may be provided and coupled to bus 502 for storing information and instructions.

[124] The computer system 500 may implement the techniques described herein using customized hard-wired logic, one or more ASICs or FPGAs, firmware and/or program logic which in combination with the computer system causes or programs computer system 500 to be a special-purpose machine. According to one embodiment, the operations, methods, and processes described herein are performed by computer system 500 in response to processor(s) 504 executing one or more sequences of one or more instructions contained in main memory 506. Such instructions may be read into main memory 506 from another storage medium, such as storage device 510. Execution of the sequences of instructions contained in main memory 506 may cause processor(s) 504 to perform the process steps described herein. In alternative

embodiments, hard-wired circuitry may be used in place of or in combination with software instructions.

[125] The main memory 506, the ROM 508, and/or the storage device 510 may include non-transitory storage media. The term “non-transitory media,” and similar terms, as used herein refers to media that store data and/or instructions that cause a machine to operate in a specific fashion, the media excludes transitory signals. Such non-transitory media may comprise non-volatile media and/or volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 510. Volatile media includes dynamic memory, such as main memory 506. Common forms of non-transitory media include, for example, a floppy disk, a flexible disk, hard disk, solid state drive, magnetic tape, or any other magnetic data storage medium, a CD-ROM, any other optical data storage medium, any physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, NVRAM, any other memory chip or cartridge, and networked versions of the same.

[126] The computer system 500 may include a network interface 518 coupled to bus 502. Network interface 518 may provide a two-way data communication coupling to one or more network links that are connected to one or more local networks. For example, network interface 518 may be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, network interface 518 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN (or WAN component to communicate with a WAN). Wireless links may also be implemented. In any such implementation, network interface 518 may send and receive electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[127] The computer system 500 can send messages and receive data, including program code, through the network(s), network link and network interface 518. In the Internet example, a server might transmit a requested code for an application program through the Internet, the ISP, the local network and the network interface 518.

[128] The received code may be executed by processor(s) 504 as it is received, and/or stored in storage device 510, or other non-volatile storage for later execution.

[129] Each of the processes, methods, and algorithms described in the preceding sections may be embodied in, and fully or partially automated by, code modules executed by one or more

computer systems or computer processors comprising computer hardware. The processes and algorithms may be implemented partially or wholly in application-specific circuitry.

[130] The various features and processes described above may be used independently of one another or may be combined in various ways. All possible combinations and sub-combinations are intended to fall within the scope of this specification. In addition, certain method or process blocks may be omitted in some implementations. The methods and processes described herein are also not limited to any particular sequence, and the blocks or states relating thereto can be performed in other sequences that are appropriate. For example, described blocks or states may be performed in an order other than that specifically disclosed, or multiple blocks or states may be combined in a single block or state. The examples of blocks or states may be performed in serial, in parallel, or in some other manner. Blocks or states may be added to or removed from the disclosed embodiments. The examples of systems and components described herein may be configured differently than described. For example, elements may be added to, removed from, or rearranged compared to the disclosed embodiments.

[131] The various operations of methods described herein may be performed, at least partially, by one or more processors that are temporarily configured (e.g., by software) or permanently configured to perform the relevant operations. Whether temporarily or permanently configured, such processors may constitute processor-implemented engines that operate to perform one or more operations or functions described herein.

[132] Similarly, the methods described herein may be at least partially processor-implemented, with a particular processor or processors being an example of hardware. For example, at least some of the operations of a method may be performed by one or more processors or processor-implemented engines. Moreover, the one or more processors may also operate to support performance of the relevant operations in a “cloud computing” environment or as a “software as a service” (SaaS). For example, at least some of the operations may be performed by a group of computers (as examples of machines including processors), with these operations being accessible via a network (e.g., the Internet) and via one or more appropriate interfaces (e.g., an Application Program Interface (API)).

[133] The performance of certain of the operations may be distributed among the processors, not only residing within a single machine, but deployed across a number of machines. In some embodiments, the processors or processor-implemented engines may be located in a single

geographic location (e.g., within a home environment, an office environment, or a server farm). In other embodiments, the processors or processor-implemented engines may be distributed across a number of geographic locations.

[134] Throughout this specification, plural instances may implement components, operations, or structures described as a single instance. Although individual operations of one or more methods are illustrated and described as separate operations, one or more of the individual operations may be performed concurrently, and nothing requires that the operations be performed in the order illustrated. Structures and functionality presented as separate components in configurations may be implemented as a combined structure or component. Similarly, structures and functionality presented as a single component may be implemented as separate components. These and other variations, modifications, additions, and improvements fall within the scope of the subject matter herein.

[135] Although an overview of the subject matter has been described with reference to specific embodiments, various modifications and changes may be made to these embodiments without departing from the broader scope of embodiments of the specification. The Detailed Description should not to be taken in a limiting sense, and the scope of various embodiments is defined only by the appended claims, along with the full range of equivalents to which such claims are entitled. Furthermore, related terms (such as “first,” “second,” “third,” etc.) used herein do not denote any order, height, or importance, but rather are used to distinguish one element from another element. Furthermore, the terms “a,” “an,” and “plurality” do not denote a limitation of quantity herein, but rather denote the presence of at least one of the articles mentioned.