Dynamic Recognition of Speakers for Consent Management by Contrastive Embedding Replay

Arash Shahmansoori and Utz Roedig, *Member, IEEE*

***Abstract*— Voice assistants overhear conversations, and a consent management mechanism is required. Consent management can be implemented using speaker recognition. Users that do not give consent enroll their voice, and all their further recordings are discarded. Building speaker recognition-based consent management is challenging as dynamic registration, removal, and reregistration of speakers must be efficiently handled. This work proposes a consent management system addressing the aforementioned challenges. A contrastive-based training is applied to learn the underlying speaker equivariance inductive bias. The contrastive features for buckets of speakers are trained a few steps into each iteration and act as replay buffers. These features are progressively selected using a multi-strided random sampler for classification. Moreover, new methods for dynamic registration using a portion of old utterances, removal, and reregistration of speakers are proposed. The results verify memory efficiency and dynamic capabilities of the proposed methods and outperform the existing approaches from the literature in terms of convergence rate and number of required parameters.**

***Index Terms*— Consent management, contrastive embedding replay, dynamic learning, multi-strided sampling, voice assistant systems.**

I. Introduction

MANY recent Internet of Things (IoT) applications, such as smart homes, smart transport systems, or smart healthcare, rely on voice assistants as primary user interface. This is due to the fact that end users prefer to communicate with IoT devices more naturally, using voice commands rather than classical interfaces, such as a touch screen [1]. Consequently, consent management is now becoming a concern. For example, the recent European Union legislation, general data protection regulation (GDPR), requires all parties' consent for personal data collection. In the context of voice assistant systems, providing this feature is essential to protect users from being recorded without giving consent. If not giving consent, users should at least be able to communicate dissent, such that their voice is not recorded. Implementing such a consent/dissent management system for voice assistants is challenging. The existence of voice assistant systems to nearby users may initially not be evident. Also, there is no obvious interface to articulate consent or dissent.

Recent initial ideas to implement consent management can be divided in two broad categories: the consent management without and with voice assistant support. The first category assumes that the operators of a voice assistant ecosystem do not support the implementation of consent management, while the second approach assumes collaboration of a provider, e.g., Amazon in the case of the Echo voice assistant. In the first category, denial of service approaches have been proposed; the voice assistant is prevented to collect voice samples by a nonconsenting party. Specifically, an acoustic jamming device can be used to prevent all voice assistant systems in the vicinity of a user to record [2]. While such approach is possible, it is difficult to implement reliably in practical settings. In the second category, more options are available. One approach is to add information to the acoustic channel that can subsequently be detected by a back end. A sound signal, i.e., a tag, is embedded in the audio stream via a speaker that can be used by the voice assistant's back end for consent management [3]. This approach faces challenges, in particular when consent of multiple users should be handled, requiring collision management of tag signals.

A second approach in this category is the use of speaker recognition for consent management. However, the direct use of such approaches in the context of consent management is not practical as will be briefly discussed. In [4], [5], and [6], few-shot learning methods are used to generalize on the classes with similar features never seen during the training mode for speaker recognition. However, in the context of consent management for voice assistant systems, such a generalization actually hurts the consent management as a privacy measure. This is due to the fact that there is a possibility for generalizing to speakers that are already providing their consent according to the samples from the speakers that do not. In [7], [8], and [9], replay-based buffer methods for continually learning a set of tasks are proposed, such that each time the network only has full access to the data for the current task. However, these approaches usually require difficult ways to generate the replay, learn the parameters of a target network, and sample the buffer in the input space leading to slow convergence,

Manuscript received 6 February 2023; revised 3 July 2023; accepted 11 September 2023. This work was supported by the Science Foundation Ireland under Grant 19/FFP/6775. *(Corresponding author: Arash Shahmansoori.)*

The authors are with the School of Computer Science and Information Technology, University College Cork, Cork, T12 XF62 Ireland (e-mail: arash.mansoori65@gmail.com; u.roedig@cs.ucc.ie).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3317493.

Digital Object Identifier 10.1109/TNNLS.2023.3317493

performance degradation, computationally complex operations, and large memory requirements. Moreover, it is assumed that the entire data for each task are provided sequentially, and the network is fully trained for the current task using the replay buffer of the previous tasks to avoid catastrophic forgetting. This is not necessarily the case for the consent management systems, as generally only a small portion of dataset for each bucket of speakers may be provided during each iteration.

In [10], [11], [12], [13], [14], and [15], different methods are applied for speaker verification systems. Such applications usually require large batch size, larger models, and full access to the entire utterances of speakers during training. However, this leads to slow convergence, large memory requirements, and performance degradation with partial access to utterances of speakers for consent management in voice assistant systems. Moreover, the main concern for speaker verification systems is the existence of speakers in a pool of previously registered speakers. This can be useful for certain applications that only need to screen a set of speakers and verify their existence [16].

In the context of consent management for voice assistant systems, the main concerns are about the dynamic management of consent for "the specific speakers" in "the specific buckets," efficient use of their utterances, and not storing their private information unnecessarily in the back end during new registrations. Moreover, it is totally possible for speakers not to provide their consent for certain attributes, e.g., gender, but providing their consent for other attributes, e.g., transcribing their speech. In other words, "identifying" the speakers who do not provide consent is of particular interest, as they may provide their consent for certain attributes. In conclusion, it is not a zero-sum game to verify the existence of speakers or screen a given set of speakers, but rather a dynamic process to manage their consent and identify them.

The specific contributions are summarized as follows.

1) A training process based on the contrastive embeddings as a way to learn speaker equivariance inductive bias is proposed. The proposed approach is efficient in terms of convergence speed and accurate prediction of speakers that do not provide consent. This is mainly due to learning the underlying speaker equivariance inductive biases and using them as replay buffer continuously during the training for classification.
2) A progressive multi-strided random sampling of the contrastive embedding replay buffer is proposed. The proposed sampling strategy starts with the large number of utterances from the initial buckets to fill up the memory size. Then, it sparsely samples the buckets of speakers to preserve enough memory for the buckets seen so far. This leads to memory efficiency, progressive increase of task difficulty, and avoiding parameter shift to the buckets of speakers with more samples.
3) A dynamic algorithm for registering new speakers in different buckets is proposed. The new speakers are registered, using only a portion of the utterances of old speakers, in the unique buckets, obtained according to $L2$ pairwise distance from the prototypes of the previous registrations, in each round. This is achieved using a dynamic programming with linear time complexity.

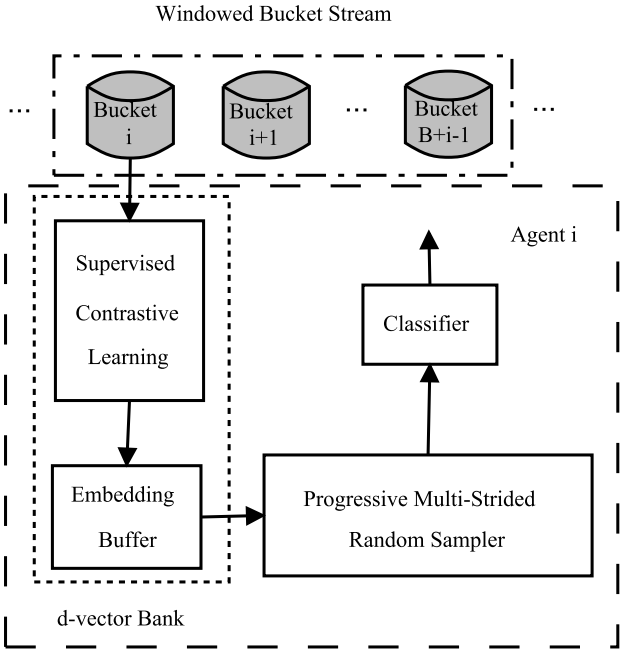

Fig. 1. Process for the proposed training with contrastive embedding replay for an agent.

4) A dynamic algorithm for removing the previously registered speakers from the pool of speakers is proposed. The proposed algorithm is capable of selectively forgetting the previously learned contrastive features for speakers in different buckets with the reduced elapsed time. Also, the proposed method can quickly reregister the removed speakers in case this is required.
5) All the aforementioned points are applied for both supervised and unsupervised modes. In particular, the proposed method for the supervised contrastive learning is explained, and the results for both supervised and unsupervised cases are provided in the experiments.

II. Method

In this section, first, the framework for training of speakers for consent management is explained in an algorithmic way. Then, a mechanism for dynamic registration of new speakers' consent is proposed. Finally, a method for removing the previously registered speakers' consent is developed.

A. Training

The framework for the entire training process is described by an agent interacting with groups of speakers, i.e., buckets. Each agent is responsible for training a windowed stream of buckets in a modular manner. This way, it is possible to distribute the training process among different agents. Fig. 1 shows the overall pictorial viewpoint of the proposed training with contrastive embedding replay in the supervised mode. The extension of the proposed approach to the unsupervised mode is provided in the results.

The proposed method starts the training with the selection of buckets of speakers that do not provide consent, i.e., a windowed bucket stream, for each agent. Subsequently, each

bucket of speakers is fed to a supervised contrastive learning framework where feature extraction is achieved by running the supervised contrastive learning for each bucket individually only a few steps into the training. Note that contrastive feature extraction in each run of the training process requires only a few steps, e.g., $\text{epochs}_{\text{cont}} = 5$. In other words, the proposed method does not wait for the full convergence of the supervised contrastive learning for each bucket during each run of the training process. The individual supervised contrastive loss $\mathcal{L}_{\text{sup}}^{(b)}$ for bucket $b \in \mathbf{b}_i$, where $\mathbf{b}_i = [i, i+1, \ldots, B+i-1]$ denotes the list of B buckets for the agent i, is defined as follows:

$$\mathcal{L}_{\text{sup}}^{(b)} = \sum_{(s,u)\in\mathcal{I}_b} \mathcal{L}_{\text{sup},s,u}^{(b)} \tag{1}$$

where $\mathcal{I}_b$ denotes all the speakers' utterances in the batch for the bucket b during training and $\mathcal{L}_{\text{sup},s,u}^{(b)}$ is defined as follows:

$$\mathcal{L}_{\text{sup},s,u}^{(b)} = \frac{-1}{\left|\mathcal{P}_{s,b}(u)\right|} \times \sum_{p\in\mathcal{P}_{s,b}(u)} \log\left(\frac{\exp\left(\mathbf{z}_{s,b}^{(u)} \cdot \mathbf{z}_p/\tau\right)}{\sum_{a\in\mathcal{A}_{s,b}(u)} \exp\left(\mathbf{z}_{s,b}^{(u)} \cdot \mathbf{z}_a/\tau\right)}\right) \tag{2}$$

where $\mathcal{A}_{s,b}(u)$ and $\mathcal{P}_{s,b}(u)$ are defined as follows:

$$\mathcal{A}_{s,b}(u) := \mathcal{I}_{s,b} \backslash \{u\} \tag{3}$$

$$\mathcal{P}_{s,b}(u) := \mathcal{P}_{s,b} \backslash \{u\} \tag{4}$$

in which $\mathcal{I}_{s,b}$ and $\mathcal{P}_{s,b}$ denote all the utterances of other speakers $\tilde{s} \neq s$ in the bucket b, and all the utterances of speaker s in the bucket b, respectively. The operation $\backslash\{u\}$ excludes the anchor utterance u from the corresponding set, and $|\mathcal{P}_{s,b}(u)|$ denotes the corresponding cardinality of the set of utterances for speaker s in bucket b excluding anchor utterance u. The parameter τ is a positive scalar denoting the temperature. The embedding terms in (2) are obtained as follows:

$$\mathbf{z}_{s,b} = \text{Proj}_{\theta_{\text{proj},b}}\left(\text{Emb}_{\theta_{e,b}}\left(\mathbf{x}_{s,b}\right)\right) = \text{Enc}_{\theta_b}\left(\mathbf{x}_{s,b}\right) \tag{5}$$

where $\text{Emb}_{\theta_{e,b}}(\cdot)$ and $\text{Proj}_{\theta_{\text{proj},b}}(\cdot)$ denote the embedding network and the projection head, respectively. The notation $\text{Enc}_{\theta_b}(\cdot)$ is used for the encoder containing the embedding network followed by the projection head. The projection head is implemented using the attention pooling layer to obtain the embeddings for speaker s in bucket b, $\mathbf{z}_{s,b}$, with the parameter set $\theta_b = \{\theta_{e,b}, \theta_{\text{proj},b}\}$. The embedding $\mathbf{z}_{s,b}$ contains the elements $\mathbf{z}_{s,b}^{(j)}$ for the jth utterance, $\mathbf{z}_p/\mathbf{z}_a$ denotes the corresponding positive/negative embedding, and $\mathbf{x}_{s,b}$ denotes the input features obtained as described in the simulations.

Subsequently, the contrastive embedding buffer is sampled according to a progressive multi-strided random sampling algorithm, described by a collection of functions in Appendix A. Finally, the classifier is trained using the samples provided by the aforementioned progressive multi-strided random embedding buffer sampling algorithm. In other words, the contrastive training provides an inductive bias for speaker classification during training as the main task. Presenting the contrastive inductive bias to the main classification task during training results efficient use of data and fast convergence as will be discussed in the simulations. Fig. 2 represents the proposed method after training in the inference mode. In this mode, utterances of unknown bucket of unknown speaker(s) are provided to the trained agent. Using the supervised contrastively trained feature extraction, a bank of d vectors is achieved that can be used as the inputs to the trained classifier for inferring the speaker(s) together with the corresponding bucket. To simplify the notation, the subscript i for the bucket list $\mathbf{b}_i$ is dropped for the rest of the manuscript.

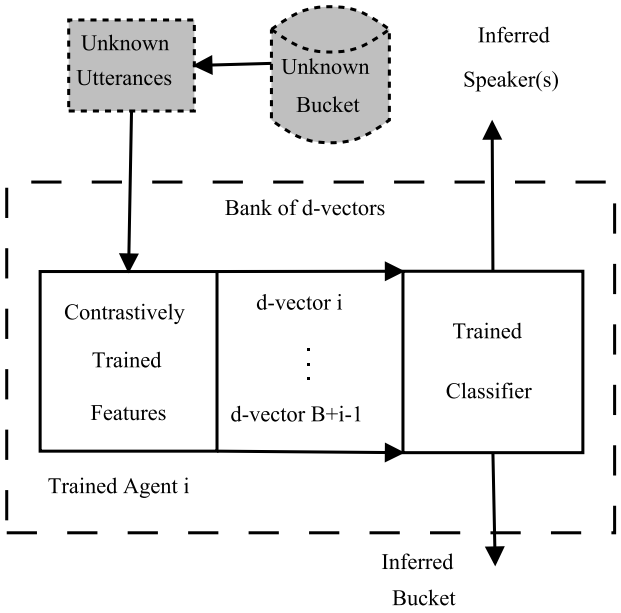

Fig. 2. Pictorial viewpoint of the proposed method in the inference mode for a given agent.

The entire process of consent management is proposed in Algorithm 1. After initializing the parameters of contrastive feature extraction encoder $\text{Enc}_{\{\theta_b\}}(\cdot)$, $\{\theta_b\}$, and classifier $\text{Cls}_\phi(\cdot)$, ϕ, the list $\mathbf{n}_b^{\text{reg}} \in \{0,1\}^B$ containing the number of new speakers per buckets, with the values selected from the set $\{0, 1\}$, as will be explained in dynamic registration procedure, is set to zero. This is due to the fact that there are no new registered speakers in the pool of speakers for consent management. In step 1, the number of utterances per speaker $n_{\text{spk,utt}}$ is obtained as follows:

$$n_{\text{spk,utt}} = \text{num}_{\text{spk,utts}}\left(\text{max}_{\text{mem}}, \mathbf{n}_b, \mathbf{n}_b^{\text{reg}}\right) \tag{6}$$

where the function $\text{num}_{\text{spk,utts}}(\cdot)$, defined in Appendix A, computes $n_{\text{spk,utt}}$ according to the maximum allowed memory size max_{mem} for training. The argument $\mathbf{n}_b \in \mathbb{Z}_{\geq 0}^B$ denotes a list of length B containing a non-negative set of integers representing the number of speakers per buckets. The training iterations over the specified range of epochs start at step 2. Prior to starting the registration of speakers in the buckets, a dictionary, with the keys of bucket b and values of the flattened list of indices of utterances per speakers per bucket b, is obtained in step 3 as follows:

$$\mathcal{C}_{\text{indx}} = \text{collection}_{\text{indx}}\left(n_{s,\text{utt}}, n_{\text{spk,utt}}, \mathbf{n}_b, \mathbf{n}_b^{\text{reg}}\right) \tag{7}$$

Algorithm 1 Consent Management With Contrastive Embedding Replay

1 Compute $n_{\text{spk,utt}}$ according to equation (6).
2 **for** *epoch in range*(*epochs*) **do**
3 Obtain $\mathcal{C}_{\text{indx}}$ according to equation (7).
4 $\mathbf{zy}_{\text{init}} = ([], [])$
5 **for** _, b *in enumerate*(**b**) **do**
6 Load a random shard of dataset for speakers in b with n_{utt}.
7 **if** *!(early-stop$_b$)* **then**
8 Train $\text{Enc}_{\theta_b}(\cdot)$ for $epochs_{\text{cont}}$ contrastively and save checkpoints.
9 **end**
10 Return the embeddings in equation (5) for latest checkpoints and corresponding labels.
11 $\mathbf{Z}_{\text{buff}}^{max_{\text{mem}}}, \mathbf{y}_{\text{buff}}^{max_{\text{mem}}} = sample_{\text{int-bkt}}(\mathcal{C}_{\text{indx}}[b], \mathbf{zy}_b, \mathbf{zy}_{\text{init}})$
12 Train $\text{Cls}_\phi(\cdot)$ using $\{\mathbf{Z}_{\text{buff}}^{max_{\text{mem}}}, \mathbf{y}_{\text{buff}}^{max_{\text{mem}}}\}$ for $epochs_{\text{cls}}$ and save checkpoints.
13 **end**
14 Progressively evaluate $eval_{\text{metric},b}(.)$ for $b \in \mathbf{b}$.
15 Update "early-stop$_b$" parameters according to $eval_{\text{metric},b}(.)$.
16 **if** *early-stop*$_{\mathbf{b}[-1]}$ **then**
17 Break the training.
18 **end**
19 **end**

where the function $\text{collection}_{\text{indx}}(\cdot)$ is defined in Appendix A and $n_{s,\text{utt}}$ denotes the number of selected utterances per speaker. The iterations over **b** start in step 5 where the enumerate(·) generates a specific bucket b for each iteration. In step 6, a random shard of dataset for n_{utt} utterances per speakers in b for each epoch is loaded. In case the early stopping status obtained according to the progressive evaluation of a given metric, e.g., accuracy or loss, up to bucket $b \in \mathbf{b}$ is not "true," denoted as !(early-stop$_b$), where !(·) negates the logical statement in parenthesis, train $\text{Enc}_{\theta_b}(\cdot)$ for $\text{epochs}_{\text{cont}}$ steps contrastively, shown in steps 7–9. This is due to the fact that the task difficulty is progressively increased during each iteration. In particular, the number of speakers is increased by providing samples from each bucket progressively, and the number of utterances per speaker is decreased as a result of the max_{mem} memory budget. In other words, if the classifier is able to distinguish different classes with sufficiently high accuracy for harder tasks, according to the corresponding contrastively trained features, it is also able to classify the simpler tasks prior to that task.

Next, the speaker embeddings are obtained according to (5) for the latest available checkpoints, and the corresponding labels are returned for b in step 10. Using the interbucket sampling function $\text{sample}_{\text{int-bkt}}(\cdot)$, described in Appendix A, the progressive features $\mathbf{Z}_{\text{buff}}^{\text{max}_{\text{mem}}}$ and corresponding labels $\mathbf{y}_{\text{buff}}^{\text{max}_{\text{mem}}}$ with max_{mem} memory size are obtained in step 11. The progressive features and the corresponding labels are used to train $\text{Cls}_\phi(\cdot)$ in step 12 for $\text{epochs}_{\text{cls}}$ steps, and the corresponding checkpoints are saved. After the completion of iterations over $\tilde{\mathbf{b}}$, i.e., steps 5–13, the metrics, e.g., accuracy and loss, are evaluated progressively using $\text{eval}_{\text{metric},b}(\cdot)$ for $b \in \mathbf{b}$ for the holdout utterances in each epoch in step 14. Subsequently, the parameters of early-stop$_b$, e.g., the internal counter, score, and status, are updated according to the progressive metrics from the previous step in step 15. In case the hardest progressive task has a "true" early stopping status, the training will be stopped as described in steps 16–18. The hardest progressive task is the task after registering the last bucket $\mathbf{b}[-1]$ with the largest number of classes, i.e., the total number of speakers N, and the fewest utterances per speaker, due to the limited allowed memory of max_{mem}.

B. Dynamic Registration of Speakers' Consent

The dynamic registration of new speakers' consent to the pool of previously registered buckets of speakers is described in this section. For the dynamic process of registering new speakers' consent, it is required to optimally allocate the Euclidean space for new speakers. To this end, the shortest $L2$ pairwise distance is used as a metric to find the optimal buckets for new speakers; see Appendix B. In other words, registering new speakers into the buckets with shortest $L2$ pairwise distance requires less Euclidean space. Consequently, it is possible to register more new speakers in the disjoint updated feature space of the buckets. This property is essential for bucket prediction that requires disjoint buckets in the feature space. According to the above explanations, the buckets with the shortest $L2$ pairwise distance from the new speakers are referred to as optimal buckets in this article.

As the number of buckets is usually smaller than the number of new speaker registrations, there are at least two new speakers registered in the same bucket during each iteration. However, the registration of a new speaker in a bucket changes the contrastive feature state of that bucket, such that it may no longer be the optimal bucket for registering the subsequent new speaker. As a result, new speakers in the subsequent round may select different optimal buckets, according to the shortest $L2$ pairwise distance, after registrations of new speakers in the current round.

Fig. 3 shows the overall process for the dynamic registration of new speakers for two subsequent rounds. The process in the evaluation mode is shown by dotted line, while the registration itself is shown by solid line. The proposed approach starts dynamic registration by finding optimal buckets and the corresponding new speakers in the evaluation mode for each round. This is achieved using opt_spk_bkt(·) (see Appendix C)

$$\mathbf{b}_{\text{reg}}^*, \mathbf{s}_{\text{reg}}^*, \mathbf{b}_{\text{sofar}}^*, \mathbf{s}_{\text{sofar}}^* = \text{opt\_spk\_bkt}\left(\bar{\mathbf{z}}^{\text{eval}}, \mathbf{b}, \mathbf{s}_{\text{reg}}, n_{\text{round}}\right) \quad (8)$$

where $\mathbf{b}_{\text{reg}}^*$ and $\mathbf{s}_{\text{reg}}^*$ denote unique optimal buckets and corresponding new speakers to be dynamically registered for the round n_{round}. The terms $\mathbf{b}_{\text{sofar}}^*$ and $\mathbf{s}_{\text{sofar}}^*$ represent optimal buckets and corresponding new speakers that are already registered so far, i.e., prior to the round n_{round}. In (8), the

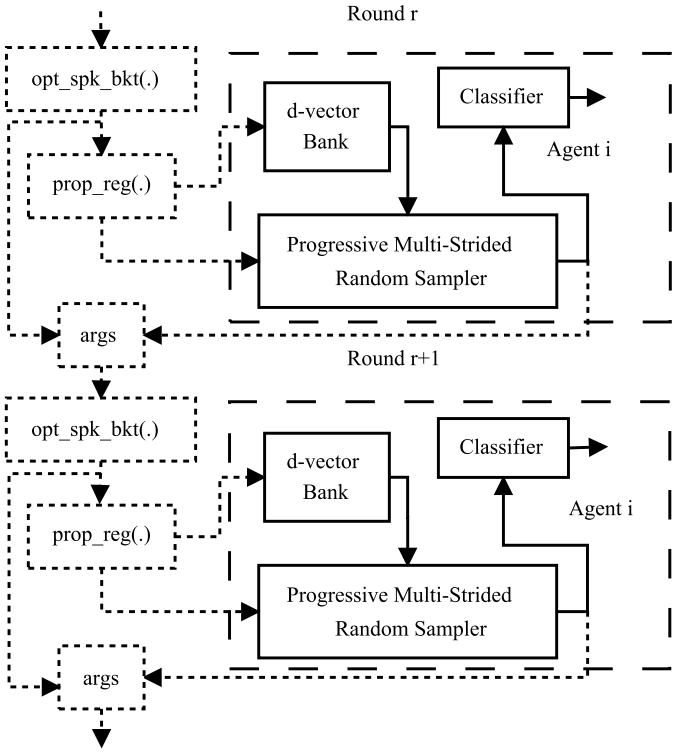

Fig. 3. Process for the proposed dynamic registration of new speakers. The evaluation mode is shown by dotted line, and the registration process is shown by solid line. The optimal buckets/speakers are obtained by opt_spk_bkt(·). The properties for registrations are obtained by prop_reg(·) and used for d-vector contrastive training and progressive multi-strided sampling. For the next round, the latent features and optimal buckets/speakers of current round are used as arguments for next round using args.

function arguments $\bar{\mathbf{z}}^{\text{eval}}$ and $\mathbf{s}_{\text{reg}}$ denote a tuple of speaker embeddings in the evaluation mode according to the previously registered speakers and new speakers, and the list of new speakers to be registered, respectively. This is achieved using the args block in Fig. 3.

For the initial round, i.e., the new registration round $n_{\text{round}} = 0$, $\tilde{\mathbf{s}}_{\text{reg}}$ is set to an empty list [], and the list of new speakers to be registered $\mathbf{s}_{\text{reg}}$ is initialized as $[N, \ldots, N + N_{\text{reg}} - 1]$, where N and N_{reg} are the number of old and new speakers, respectively. The list containing the number of speakers per buckets for dynamic registrations $\tilde{\mathbf{n}}_b$ is set to the initial state $\mathbf{n}_b$, i.e., a list containing the old number of speakers per buckets prior to dynamic registrations. The latest available checkpoints of $\text{Enc}_{\{\theta_b\}}(\cdot)$ for $\forall b \in \mathbf{b}$, and $\text{Cls}_\phi(\cdot)$ in the current round are loaded. For $n_{\text{round}} = 0$, the aforementioned checkpoints, except the last linear layer of the classifier with the output dimension of $N + N_{\text{reg}}$, are loaded from the trained network with Algorithm 1 for the old N speakers. Subsequently, the parameters for optimal buckets and corresponding new speakers are obtained as described in (8). For the next round, $\mathbf{s}_{\text{reg}}$ is updated as $\mathbf{s}_{\text{reg}} \leftarrow \mathbf{s}_{\text{reg}} \backslash \mathbf{s}^*_{\text{reg}}$. In other words, new speakers to be registered in the current round are excluded from $\mathbf{s}_{\text{reg}}$ for the next round. This is shown by an arrow from the output of the opt_spk_bkt(·) block in the current round to compute the arguments of the opt_spk_bkt(·) block in the subsequent round using the args block in Fig. 3.

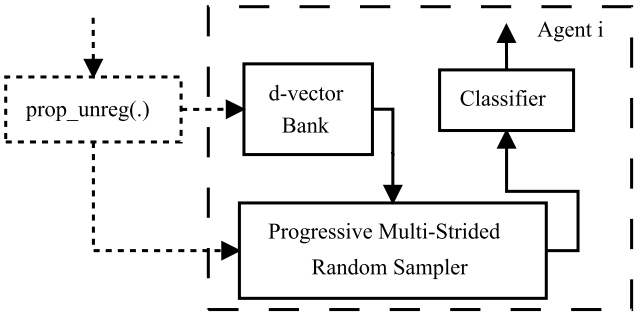

Fig. 4. Process for the proposed dynamic removal of speakers. The evaluation mode is shown by dotted line, and the removal process is shown by solid line. The properties for removal are obtained by prop_unreg(·) and used for d-vector contrastive training and progressive multi-strided sampling.

Consequently, the necessary properties for registering new speakers are obtained using prop_reg(·) defined as follows (see Appendix D):

$$\tilde{\mathbf{n}}_b, \tilde{\mathbf{n}}_b^{\text{reg}}, \mathcal{S}_{\text{reg}}, \mathcal{P}_{\text{reg}} = \text{prop\_reg}\Big(\mathbf{b}, \mathbf{n}_b, \mathbf{b}^*_{\text{reg}}, \mathbf{b}^*_{\text{sofar}}, \mathbf{s}^*_{\text{reg}}, \mathbf{s}^*_{\text{sofar}}\Big) \tag{9}$$

where $\tilde{\mathbf{n}}_b$ denotes the updated number of speakers per buckets, $\tilde{\mathbf{n}}_b^{\text{reg}}$ is the updated number of new speakers per buckets containing the values of zero or one, since at most one new speaker should be registered in each optimal bucket per round, $\mathcal{S}_{\text{reg}}$ represents a dictionary of new speakers in buckets with the keys of $b \in \mathbf{b}$ and values of new speakers per buckets, and $\mathcal{P}_{\text{reg}}$ denotes a dictionary of registration patterns in which the new speakers are registered in the buckets with the keys of $b \in \mathbf{b}$ and values of pattern status. The prop_reg(·) block provides necessary information for both the contrastive learning of d-vector bank according to $\mathcal{S}_{\text{reg}}$ and $\mathcal{P}_{\text{reg}}$ and progressive multi-strided sampler according to $\tilde{\mathbf{n}}_b$ and $\tilde{\mathbf{n}}_b^{\text{reg}}$, as shown in Fig. 3. In particular, progressive multi-strided sampler first obtains the updated number of utterances per speakers $\tilde{n}_{\text{spk,utt}}$ according to $\tilde{\mathbf{n}}_b$, $\tilde{\mathbf{n}}_b^{\text{reg}}$, and $\max_{\text{mem}}$, based on (6). Then, the functions $\text{collection}_{\text{indx}}$ and $\text{sample}_{\text{int-bkt}}$ are used to obtain progressive samples for training the classifier $\text{Cls}_\phi(\cdot)$. Moreover, the bank of d vectors for registration of new speakers in optimal buckets is contrastively updated according to the information provided by $\mathcal{S}_{\text{reg}}$ and the corresponding patterns in $\mathcal{P}_{\text{reg}}$. More specifically, as the $\text{pattern}_{1/3}$ represents registration of the new speaker(s) in the corresponding optimal bucket(s), it requires training of the contrastive feature encoder accordingly. However, the $\text{pattern}_{2/4}$ does not require training of the contrastive feature encoder, as it represents the already registered new speakers, pattern_2, or previously registered old speakers, pattern_4; see Appendix D. The same process for early stopping according to the progressive evaluation of metrics is applied as in Algorithm 1. Finally, Algorithm 2 summarizes different steps for the dynamic registration of new speaker(s) to the previously registered buckets of speakers.

C. Dynamic Removal of Speakers' Consent

The process for removing the previously registered speakers from the buckets is shown in Fig. 4. The process in evaluation mode is shown by dotted line, while the removal process is

Algorithm 2 Dynamic Consent Management for New Speaker Registrations

Follow the steps in equations (8) & (9).
Compute $\tilde{n}_{\text{spk,utt}}$ according to equation (6).
for *epoch in range*(*epochs*) **do**
  if *len*($\mathbf{b}^{*}_{\text{reg}}$) != 0 **then**
    Obtain $\mathcal{C}_{\text{indx}}$ according to equation (7).
    $\mathbf{zy}_{\text{init}}$ = ([], [])
    for _, b *in enumerate*($\mathbf{b}$) **do**
      Load a random shard of old dataset for speakers in b with n_{utt} and $pcnt_{\text{old}}$.
      Load a random shard of new dataset according to $\mathcal{S}_{\text{reg}}[b]$ with n_{utt}.
      Combine the loaded old and new datasets from the previous steps.
      if *!(early-stop*$_b$*)*&$\mathcal{P}_{\text{reg}}[b]$ = pattern$_{1/3}$ **then**
        Train $\text{Enc}_{\theta_b}(\cdot)$ for $epochs_{\text{cont}}$ contrastively using $\mathcal{P}_{\text{reg}}[b]$ and save checkpoints.
      end
      Return the embeddings in equation (5) for latest checkpoints and corresponding labels.
      $\mathbf{Z}^{max_{\text{mem}}}_{\text{buff}}$, $\mathbf{y}^{max_{\text{mem}}}_{\text{buff}}$ = $sample_{\text{int-bkt}}$ ($\mathcal{C}_{\text{indx}}[b]$, $\mathbf{zy}_b$, $\mathbf{zy}_{\text{init}}$)
      Train $\text{Cls}_{\phi}(\cdot)$ using $\{\mathbf{Z}^{max_{\text{mem}}}_{\text{buff}}, \mathbf{y}^{max_{\text{mem}}}_{\text{buff}}\}$ for $epochs_{\text{cls}}$ and save checkpoints.
    end
    Progressively evaluate $eval_{\text{metric},b}(.)$ for $b \in \mathbf{b}$.
    Update "early-stop$_b$" parameters according to $eval_{\text{metric},b}(.)$.
    if *early-stop*$_{\mathbf{b}[-1]}$ **then**
      Break the training.
    end
  end
end

Algorithm 3 Consent Management for Removing Previously Registered Speakers

Compute the parameters based on equation (10).
Compute $\tilde{n}_{\text{spk,utt}}$ according to equation (6).
for *epoch in range*(*epochs*) **do**
  Obtain $\mathcal{C}_{\text{indx}}$ according to equation (7).
  $\mathbf{zy}_{\text{init}}$ = ([], [])
  for _, b *in enumerate*($\mathbf{b}$) **do**
    Load a random shard of dataset according to $\mathcal{S}_{\text{res}}[b]$ with n_{utt}.
    if *!(early-stop*$_b$*)*&$\mathcal{P}_{\text{unreg}}[b]$ = pattern$_1$ **then**
      Train $\text{Enc}_{\theta_b}(\cdot)$ for $epochs_{\text{cont}}$ contrastively using $\mathcal{P}_{\text{unreg}}[b]$ and save checkpoints.
    end
    Return the embeddings in equation (5) for latest checkpoints and corresponding labels.
    $\mathbf{Z}^{max_{\text{mem}}}_{\text{buff}}$, $\mathbf{y}^{max_{\text{mem}}}_{\text{buff}}$ = $sample_{\text{int-bkt}}$ ($\mathcal{C}_{\text{indx}}[b]$, $\mathbf{zy}_b$, $\mathbf{zy}_{\text{init}}$)
    Train $\text{Cls}_{\phi}(\cdot)$ using $\{\mathbf{Z}^{max_{\text{mem}}}_{\text{buff}}, \mathbf{y}^{max_{\text{mem}}}_{\text{buff}}\}$ for $epochs_{\text{cls}}$ and save checkpoints.
  end
  Progressively evaluate $eval_{\text{metric},b}(.)$ for $b \in \mathbf{b}\backslash\mathbf{b}_{\text{unreg}}$, and per bucket for $b \in \mathbf{b}_{\text{unreg}}$.
  Update "early-stop$_b$" parameters according to $eval_{\text{metric},b}(.)$.
  if *early-stop*$_{\mathbf{b}\backslash\mathbf{b}_{\text{unreg}}[-1]}$&*all(early-stop*$_{b\in\mathbf{b}_{\text{unreg}}}$*)* **then**
    Break the training.
  end
end

shown by solid line. The properties of interest for removing the given set of speakers from the pool of previously registered speakers are obtained using the function prop_unreg(·) (see Appendix E)

$$\tilde{\mathbf{n}}_b, \mathcal{S}_{\text{res}}, \mathcal{P}_{\text{unreg}} = \text{prop\_unreg}\big(\mathbf{b}, \mathbf{n}_b, \mathbf{b}_{\text{unreg}}, \mathbf{s}_{\text{res}}\big) \quad (10)$$

where $\mathbf{b}_{\text{unreg}}$ denotes the corresponding unique set of buckets for removing the given set of speakers and $\mathbf{s}_{\text{res}}$ represents the set of residual speakers after removing the given set of speakers. Consequently, the properties of interest, including the updated number of speakers per buckets $\tilde{\mathbf{n}}_b$, a dictionary of updated residual speakers in buckets $\mathcal{S}_{\text{res}}$ with the keys of $b \in \mathbf{b}$ and values of residual speakers per buckets, and a dictionary of patterns for removing the speakers with the keys of $b \in \mathbf{b}$ and values of pattern status, are used for contrastive update of d-vector bank and progressive multi-strided sampler. In particular, $\tilde{\mathbf{n}}_b$ is used to obtain the progressive samples for training the classifier, and $\mathcal{S}_{\text{res}}$ is used for updating the bank of d vectors for removing specific speakers from given buckets according to the corresponding pattern in $\mathcal{P}_{\text{unreg}}$.

Algorithm 3 summarizes different steps for the dynamic removal of speaker(s). It is worth mentioning that the proposed algorithm assumes the existence of at least two residual speakers per bucket required for contrastive training. In the results, it is explained how to deal with other cases. The early stopping status is obtained according to the progressive evaluation of all the buckets except the unique set of buckets for removing $\mathbf{b}_{\text{unreg}}$, shown as $b \in \mathbf{b}\backslash\mathbf{b}_{\text{unreg}}$, and per bucket evaluation of $b \in \mathbf{b}_{\text{unreg}}$ for a given metric, e.g., accuracy or loss. This is due to the fact that evaluation metric for the bucket(s) comprising the unregistered speaker(s) is obtained for the entire holdout utterances, including the unregistered speakers. For example, if one speaker is removed from a given bucket with five speakers, the expected metric for accuracy of that bucket is around 80%. Consequently, if the early stopping status is not true, i.e., !(early-stop$_b$), and the removal pattern follows pattern$_1$, then $\text{Enc}_{\theta_b}(\cdot)$ is trained for epochs$_{\text{cont}}$ steps contrastively according to the given pattern, steps 8–10. This is due to the fact that pattern$_1$ requires training of the contrastive features excluding the samples of the unregistered speaker(s) from the bucket for selective forgetting. On the other hand, pattern$_2$ does not require training of the contrastive features, as it is related to the bucket(s) that do not include unregistered

speaker(s); see Appendix E. In case the hardest progressive task, excluding the $\mathbf{b}_{\text{unreg}}$ and shown as early-stop$_{\mathbf{b}\setminus\mathbf{b}_{\text{unreg}}[-1]}$, with the same definition as in Algorithm 1, together with all the bucket(s) comprising the unregistered speaker(s), all(early-stop$_{b\in\tilde{\mathbf{b}}_{\text{unreg}}}$), has "true" early stopping statuses, the training will be stopped as described in steps 17–19. The process for reregistering follows a similar procedure as in Algorithm 3 by reregistering the unregistered speaker(s) in the corresponding bucket(s).

It is worth noting that the bucket index may encode information about the duration of dissent in practice. This way, speakers that do not provide consent for a given time interval are grouped in the buckets with the corresponding time stamps stored as a decision tree in the back end. Consequently, the problem boils down to a decomposable search algorithm that is known to be a fully retroactive data structure via decision trees with the overhead of $O(\log(B))$ for B buckets [17].

III. Experiments

The goal of the simulations is to answer the following questions for both supervised and unsupervised modes.

1) Can the proposed method enable a fast training on different datasets?
2) Can the proposed method dynamically register new speakers efficiently?
3) Can the proposed method dynamically remove and reregister the speakers efficiently?
4) Can the proposed method provide a good verification performance?

All the experiments were run on a single NVIDIA GeForce RTX 2070 GPU, and Python version 3.9.4 was used to implement the algorithms. The code for the simulations is available at [18].

A. Dataset

The LibriSpeech dataset is used for the results unless otherwise stated [19]. In addition, the VoxCeleb dataset is considered to compare the performance of the proposed method with the corresponding methods from the literature [20]. Different subsets of the aforementioned datasets are used for training and testing. In particular, different agents are used for the simulations each of which using $N = 40$ different speakers selected from the set of speakers with lower word error rate, denoted by "clean" in the LibriSpeech dataset, and randomly selected for the case of VoxCeleb dataset according to the agent index. In other words, for agent i, speakers $(i \times N, (i+1) \times N)$ are selected and equally divided in $B = 8$ different buckets. For the registration of new speakers in the previously trained contrastive buckets of speakers, $N_{\text{reg}} = 20$ speakers are selected from the set of speakers with higher word error rate, denoted by "other" in the LibriSpeech dataset and briefly referred to as new speakers with noisy utterances, according to the agent index. In other words, for agent i, new speakers $(i \times N_{\text{reg}} + N, (i+1) \times N_{\text{reg}} + N)$ are dynamically registered in the pool of previously registered speakers $(i \times N, (i+1) \times N)$.

TABLE I
Embedding Network per Bucket $\text{Emb}_{\theta_{e,b}}(\cdot)$

#	Layer (Type)	Output Shape	Param #
1	LSTM	[-1, 160, 128]	351,232
2	Linear	[-1, 160, 256]	33,024
3	tanh	[-1, 160, 256]	0
4	GroupNorm	[-1, 160, 256]	320
5	Attention Pooling	[-1, 256]	257
6	Normalization	[-1, 256]	0

TABLE II
Classifier Network $\text{Cls}_{\phi}(\cdot)$

#	Layer (Type)	Output Shape	Param #
1	Linear	[-1, 64]	16,448
2	ReLU	[-1, 64]	0
3	Linear	[-1, 64]	4,160
4	ReLU	[-1, 64]	0
5	Linear	[-1, N]	2,600
6	Softmax	[-1, N]	0

B. Hyperparameters and Network Architecture

The log mel-filterbank (MFB) features with the feature dimension of 40, the frame length of 25 ms, the stride of 10 ms, and the voice activity detection (VAD) of 20 dB are used as the input features $\mathbf{x}_{s,b}$ for the encoder $\text{Enc}_{\theta_b}(\cdot)$ in (5). Subsequently, the features are normalized and scaled by the mean and variance, respectively, along the time axis. Finally, the number of iterations for the contrastive feature extraction is set to $\text{epochs}_{\text{cont}} = 5$, and the number of iterations for the classifier is set to $\text{epochs}_{\text{cls}} = 2$ and $\text{epochs}_{\text{cls}} = 1$ for the supervised and unsupervised cases, respectively, as they provide optimal performance in terms of total elapsed time for training.

The per bucket embedding network $\text{Emb}_{\theta_{e,b}}(\cdot)$ in (5) is implemented according to Table I, where -1 in the output shape column denotes the batch dimension of a tensor. In particular, the long short-term memory (LSTM) layer is applied with feature dimension 40, cell dimension 128, and number of layers 3. The group-norm layer with the number of groups 4 and the number of channels set to the segmentation length of 160 is used according to [21]. For the study on VoxCeleb dataset, the group-norm layer is replaced by the layer norm in the feature extraction [22]. The same embedding network architecture was used to implement the algorithms for the unsupervised case. For the classification using the unsupervised learning, the first two layers of $\text{Cls}_{\phi}(\cdot)$ were used with the same hyperparameters, and the output layer was removed. It is worth noting that the performance, in terms of accuracy for the unsupervised learning, is obtained according to the "cosine similarity matrix" of the output features. Also, the process to obtain positive and negative samples for the unsupervised case is similar to [10]. For the study on VoxCeleb dataset, layers 3 and 4 in Table II for the implementation of $\text{Cls}_{\phi}(\cdot)$ are removed and replaced by the layer norm. To obtain the attention weights in the projection head in (5) required for the attentive pooling, the linear transformation with the

input dimension 256 and output dimension 1 with the Softmax activation is applied. Subsequently, the embedding terms $\mathbf{z}_{s,b}$ in (5) are obtained by multiplication of the attention weights from the previous step with $\text{Emb}_{\theta_{e,b}}(\mathbf{x}_{s,b})$, summation over the segmentation length, and normalizing by the Euclidean norm over the embedding dimension. For dynamic registrations, N is replaced by $N+N_{\text{reg}}$ in the last linear layer. The stochastic gradient descent (SGD) and adaptive moment estimation (Adam) optimizers are used for the supervised contrastive learning and classifications, respectively. For the unsupervised contrastive learning, both the embedding network and the latent feature classification are optimized using SGD. To evaluate the performance on VoxCeleb dataset, second-order clipped stochastic optimization (Sophia) is used as the best choice, in terms of convergence speed of training, for both the proposed and the corresponding methods from the literature [23].

C. Baseline

The performance of the proposed approach is compared with the baselines applying unsupervised and supervised contrastive learning [10], [11]. The main purpose of providing a baseline is not about comparing different architectures for contrastive learning, but to observe the effects of training with the proposed algorithms. In other words, the effects of the proposed methods on the training elapsed time, sample efficiency, and performance are of particular interest. Consequently, contrastive learning-based methods with different architectures can benefit from the proposed algorithms in terms of convergence speed, efficient sampling, and dynamic capabilities [12], [13], [14]. For all the results, the baselines and the proposed method follow a similar network structure for speaker embedding. The hyperparameters for the baselines are selected to provide comparable performance. In particular, the number of hidden nodes is set to 512 with the projection size of 256 and the three-layer LSTM as in [10]. Also, the corresponding group norm and layer norm with the same hyperparameters as the proposed method are applied for a fair comparison and analysis on the LibriSpeech and the VoxCeleb datasets, respectively.

D. Results

Fig. 5 shows the performance during testing in terms of accuracy for different agents, using the proposed method in supervised and unsupervised modes, together with the corresponding methods from the literature, with respect to the total elapsed time in minute (min) for training. It is observed that for all the agents, the proposed method breaks the training loop, by satisfying the early stopping mechanism in Algorithm 1, much faster than the corresponding methods from the literature. In particular, the methods from the literature approximately require {24.77, 19.89, 17.65, 30.03} and {55.50, 34.34, 32.39, 47.62} min to complete the training in supervised and unsupervised modes for the top to bottom plots and from left to right, respectively. It is worth mentioning that the supervised contrastive learning from the literature [11] is optimized to provide a more stable performance

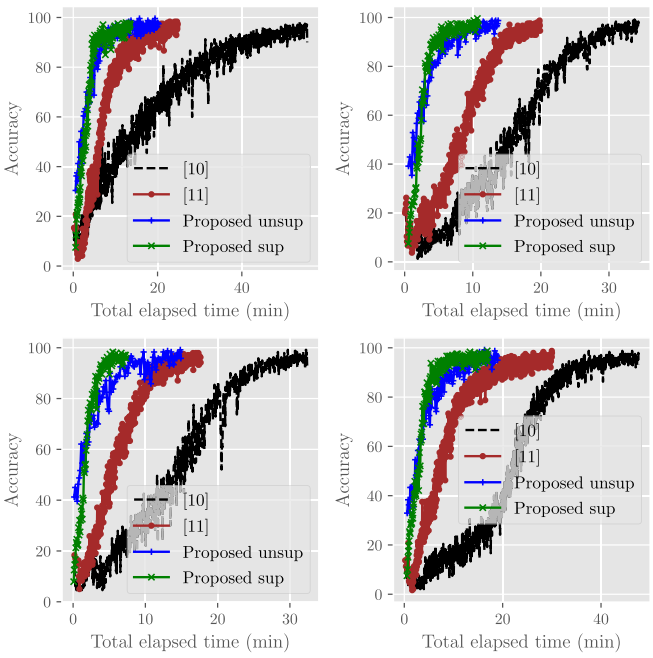

Fig. 5. Comparison between testing accuracies of different agents using the proposed contrastive embedding replay, with multi-strided progressive sampling in supervised and unsupervised modes, and the baselines with respect to the elapsed time for training.

during training. In particular, it is observed that using the mean values of latent features before applying the projection head to obtain $\mathbf{z}_p/\mathbf{z}_a$ for the corresponding positive/negative embedding, i.e., [11, eq. (2)], helps a more robust training performance for this application. More specifically, the mean values are obtained over the remaining utterances for each anchor utterance of a given speaker for positive and the corresponding utterances of the remaining speakers for negative latent features, before applying the projection head to obtain $\mathbf{z}_p/\mathbf{z}_a$, in a training batch for each iteration. The corresponding values using the proposed method are approximately {13.52, 10.82, 7.32, 17.01} and {20.21, 13.73, 14.91, 18.94} min for supervised and unsupervised modes, respectively. In other words, the required elapsed time is reduced for the agents by approximately {45.4%, 45.6%, 58.5%, 43.4%} in the supervised and {63.6%, 60%, 54%, 60.2%} in the unsupervised modes, respectively. Moreover, the baseline methods require large batch size of $20 \times N$ for $N = 40$ speakers during each iteration compared with the proposed approach that only requires $\max_{\text{mem}} = 120$. Also, the total required number of parameters in the proposed method, to provide a comparable performance with the baselines, is reduced by approximately 43% in the supervised and unsupervised modes. This is mainly related to the smaller batch size requirements and efficient use of speaker equivariance inductive bias provided by contrastive features during the training. In conclusion, the proposed approach converges much faster than the corresponding baselines due to the following points:

1) dividing different sets of speakers in the buckets;
2) contrastive learning of speaker equivariance inductive bias only a few steps into each training iteration;

TABLE III
COMPARING THE TOTAL ELAPSED TIME (MIN) FOR TRAINING ON VOXCELEB DATASET USING DIFFERENT METHODS

	Supervised		Unsupervised	
#	[11]	ours	[10]	ours
0	33.36	**12.78**	76.79	**7.19**
1	42.78	**12.61**	97.88	**9.00**
2	35.37	**8.11**	76.06	**10.23**
3	39.08	**10.19**	81.66	**8.33**
4	39.31	**10.08**	87.31	**8.74**

3) progressive increase of task difficulty by increasing number of speakers and decreasing number of utterances per speaker for a given memory budget;
4) per bucket early stopping of contrastive feature training based on the progressive evaluation of a given metric.

Table III compares the performance of the proposed method in the supervised and unsupervised modes with the corresponding methods from the literature in terms of total elapsed time in min for training. It is observed that the proposed method significantly outperforms the methods from the literature in terms of the convergence speed of training. In particular, the total elapsed time is reduced for agents 0–4 by approximately $\{61.7\%, 70.5\%, 77.1\%, 73.9\%, 74.4\%\}$ in the supervised modes and $\{90.6\%, 90.8\%, 86.6\%, 89.8\%, 90\%\}$ in the unsupervised modes, respectively. Also, the total required number of parameters in the proposed method, to provide a comparable performance with the baselines, is reduced by approximately 32% in the supervised and unsupervised modes. This is mainly due to the efficient use of contrastive inductive bias, providing easy-to-hard features using the proposed multi-strided sampling, bucketing contrastive features, and detaching the already learned contrastive features for different buckets during training. Moreover, the method [14] can benefit from the fast convergence speed of training using the proposed approach. In particular, after obtaining the pseudo-labels in the unsupervised mode, or using the labels in the semisupervised mode, the progressive multi-strided sampling and the training mechanisms proposed in this article can significantly speed up the training process. Finally, the simple framework for contrastive learning of visual representations (SimCLR)-based contrastive methods require large batch size, separate embedding networks, and increased memory usage/GPU [12], [13]. Consequently, these types of methods are not of particular interest for the specific application in this article.

Fig. 6 shows testing accuracy, for a given agent, with respect to the required elapsed time to break the dynamic registration training loop using the proposed method in Algorithm 2 for different rounds of registrations. Only 50% of the utterances of the previously registered 40 speakers are used for this simulation. The performance for different percentages of old utterances for previously registered speakers is provided subsequently. The testing accuracy of each round is reported with respect to the registration elapsed time, i.e., after breaking the training by satisfying the early stopping condition in Algorithm 2 for each round. The result is reported using different markers and colors for different rounds of registrations.

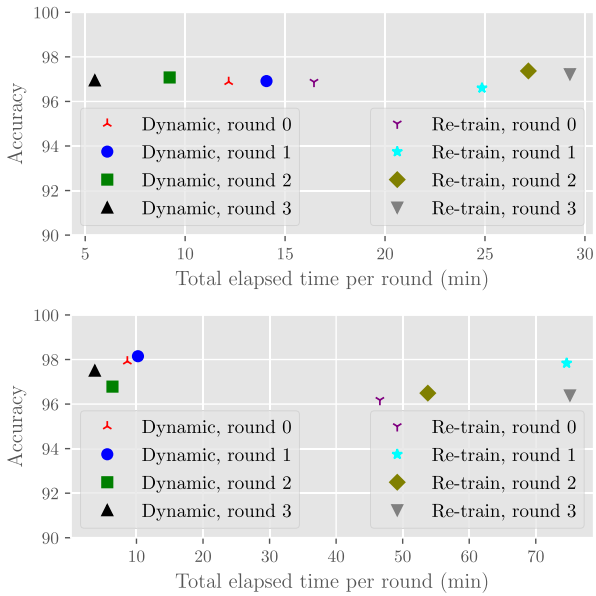

Fig. 6. Testing accuracies per round for dynamic (top) supervised and (bottom) unsupervised registrations with respect to required elapsed time to break the registration loop. Different markers and colors are used to distinguish between different rounds of dynamic registrations. The corresponding values by retraining the network per rounds are reported by different markers.

The corresponding values by retraining the network during each round, using Algorithm 1 for full utterances of old and new speakers and without the dynamic registration mechanism in Algorithm 2, are reported by different colors and markers. It is observed that the proposed dynamic registration method provides much faster registrations compared with retraining the network for each round for both supervised and unsupervised cases. In particular, by increasing the number of rounds, the total number of speakers is increased; however, due to the efficient mechanism for dynamic registrations using the information from the previous rounds of registrations, the elapsed time for subsequent registrations is decreased. On the other hand, the elapsed time of retraining the network by increasing the number of rounds is increased due to an increase in the overall number of speakers. For instance, the required elapsed times, to break the training loop, at the end of dynamic registration rounds, i.e., round 3, are approximately $\{5.5, 3.8\}$ min while it requires retraining the network for approximately $\{29, 75\}$ min for supervised and unsupervised cases, respectively.

Fig. 7 shows the testing accuracy per round for different percentages of old speaker utterances for agents 0–4, using the proposed dynamic consent management algorithm for registering new speakers in the supervised, top plot, and unsupervised, bottom plot, modes. The $\{10\%, 30\%, 50\%, 70\%, 90\%\}$ of old utterances are color coded for each agent from left to right, respectively. Moreover, different rounds of registrations are shown using different markers. It is observed that for all the agents, using pcnt $\geq$ 50% of old utterances provides the required condition for breaking the dynamic registration loop in Algorithm 2 in the supervised and unsupervised modes. However, for some agents, even using pcnt $=$ 30% of old utterances is enough to provide a similar performance, e.g.,

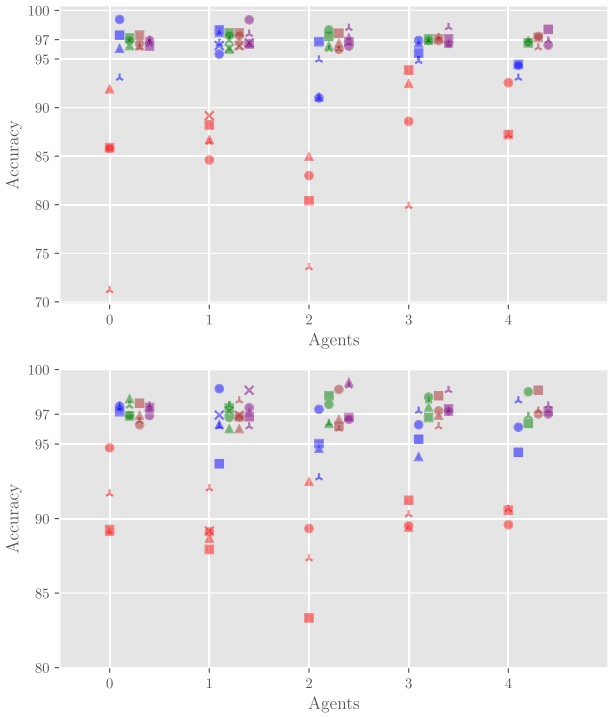

Fig. 7. Testing accuracies per round with the different percentages of old utterances for different agents in the case of dynamic (top) supervised and (bottom) unsupervised registrations. Different markers denote different rounds, and the $\{10\%, 30\%, 50\%, 70\%, 90\%\}$ percentages of old utterances are color coded for each agent from left to right, respectively. For visibility purposes, the values using different percentages of old utterances are slightly shifted to the right for each agent.

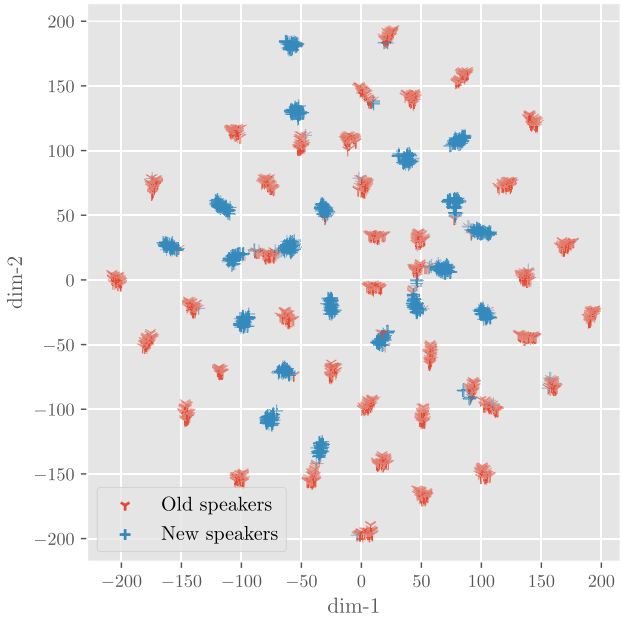

Fig. 8. Visualization of the trained latent features after the second linear layer of the classifier during testing using t-SNE. The old previously registered 40 speakers and the new dynamically registered 20 speakers are shown with different markers and colors.

agents $\{0, 1, 3\}$ in the supervised mode and agent $\{0\}$ in the unsupervised mode. The rest of agents using pcnt $=$ 30% of old utterances provide a relatively good performance for both supervised and unsupervised modes; however, their performance is slightly degraded in certain rounds. The performance starts to degrade by using only pcnt $=$ 10% of old utterances for all the agents and all rounds. This is, in particular, due to the parameter shift toward the new speaker utterances. Using a portion of old utterances during dynamic registrations is extremely useful, as the old utterances are not kept unnecessarily in the back end during new registrations, hence providing improved privacy. In other words, the proposed dynamic registration strategy provides efficient use of data from the old speakers, such that only a portion of the old utterances are required during the registrations of the new speakers without sacrificing the performance leading to improved privacy. Finally, it is possible to apply different hyperparameter optimizations and choose different values for metrics to update early stopping counter and achieve higher testing accuracy. These points are not the main purpose of this work.

Fig. 8 shows the t-distributed stochastic neighbor embedding (t-SNE) for the dynamically trained latent features after the second linear layer of the classifier during the testing [24]. It is observed that the separation between latent features of different speakers is almost perfect for the old speakers, new registrations, and among old and new features as shown by different colors and markers. In particular, the new registrations are distributed in different regions of the Euclidean space, and they are separable from the old speakers in different buckets. It is worth mentioning that the number of new registrations for each agent is upper bounded according to the limitations imposed by the Euclidean space, i.e., it is not possible to register arbitrarily large number of new speakers in each agent. Consequently, it is recommended to either create new agents or distribute new registrations between multiple agents in this case.

Fig. 9 shows the testing accuracy with respect to the required elapsed time for removal from and reregistration to a given bucket, e.g., in this case, bucket 4. The speaker(s) [20], [20, 21], and [20, 21, 22] are efficiently removed from and reregistered to bucket 4 using the proposed method. As the performance is measured on the testing utterances for all the five speakers in the bucket, the testing accuracy drops by $\{20\%, 40\%, 60\%\}$ after removing one, two, and three speakers from the bucket, respectively. In other words, Algorithm 3 efficiently loads the already trained checkpoints for feature extraction of all the other buckets and the bucket for removing/reregistering speakers together with the corresponding checkpoints for the classifier. This leads to fast convergence and breaking the contrastive training for extraction of speaker equivariance inductive bias especially for the remaining buckets. In particular, removing and reregistering the aforementioned speaker(s) require approximately $\{1.71, 1.88, 1.91\}$ min for removing and $\{1.6, 2.7, 3.95\}$ min for reregistering the speaker(s) [20], [20, 21], and [20, 21, 22], respectively, for the supervised case. For the case of unsupervised removal and reregistration, the elapsed times are approximately $\{1, 2, 2.5\}$ min for removing the speaker(s) [20], [20, 21], and [20, 21, 22] and $\{2.5, 2.8, 3.3\}$ min for

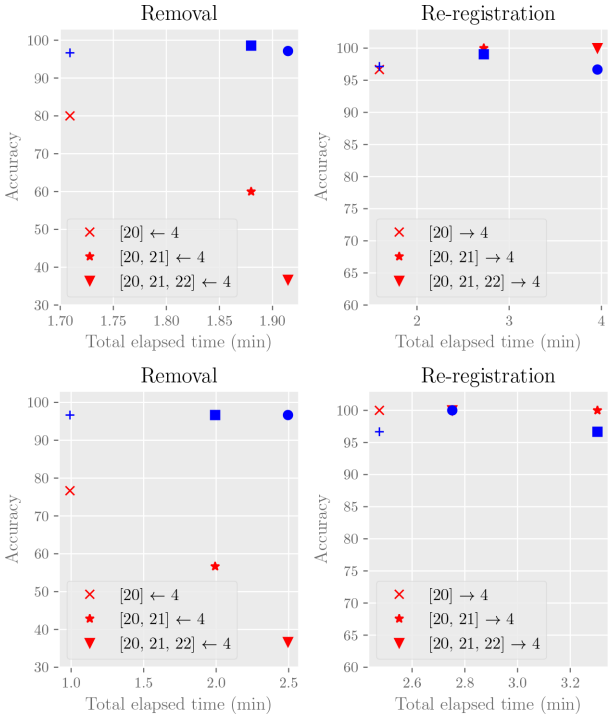

Fig. 9. Testing accuracies with respect to total elapsed time per removal/reregistration for (top) supervised and (bottom) unsupervised. The accuracies for speakers in the bucket to be removed/reregistered are displayed in a different color and different markers. For visibility purposes, the total accuracy of the remaining buckets is displayed with a different color and different markers.

reregistering the speaker(s) [20], [20, 21, 22], and [20, 21], respectively.

For the case of removing the entire speakers from the bucket, it is sufficient not to use the checkpoints of the trained contrastive feature encoder of that bucket and proceed the training without providing the data from that bucket according to Algorithm 3. This results forgetting the contrastive inductive bias of the speakers in the bucket after approximately 2.4 min and, consequently, protecting them against reidentification. Reregistering the removed bucket takes approximately 2.6 min. For the case of removing four speakers from the bucket, it is possible to reregister the remaining speaker in another bucket if available or another agent with available bucket. This leads to forgetting the inductive bias of the four speakers in the bucket by not providing the data and the corresponding checkpoints of that bucket for those speakers during the training, and absorbing the remaining speaker in another bucket. Consequently, the problem is reduced to the case that all the speakers in the bucket are removed.

Table IV reports the verification performance for different agents, i.e., agents 0–4, in terms of equal error rate (EER) in percentage, minimum of the normalized detection cost function (minDCF), and minimum cost of log likelihood ratio calibration, minC_{llr}, during the testing phase for the supervised and the unsupervised methods. The performance is reported after the completion of training procedure in Algorithm 1 for totally different holdout utterances during the test time. To analyze the verification performance in the testing phase, the entire test samples are used. It is observed that the

TABLE IV
VERIFICATION PERFORMANCE IN TERMS OF EER, minDCF, AND minC_{llr}

	Supervised			Unsupervised		
#	EER (%)	minDCF	minC_{llr}	EER (%)	minDCF	minC_{llr}
0	0.983	0.075	0.041	1.068	0.118	0.053
1	0.727	0.063	0.036	1.239	0.134	0.051
2	0.855	0.095	0.037	1.239	0.122	0.05
3	0.855	0.078	0.038	0.983	0.113	0.047
4	1.154	0.117	0.05	1.45	0.175	0.071

supervised mode always outperforms the unsupervised mode in terms of verification capabilities. This has to do with the additional information provided by the labels during the training. Moreover, the efficient use of the labeled data leads to a generally faster convergence and a better generalization capability during the inference and, hence, a better verification performance.

IV. Conclusion

In this article, an efficient method for consent management of speakers in the context of voice assistant systems is proposed. The proposed algorithms significantly reduce the convergence time of speaker recognition for consent management and outperform the baselines. Moreover, the proposed approach dynamically adapts to the consent status of each speaker. In other words, the process for registering new speakers, removing from the pool of registered speakers, and reregistering the speakers during the consent management is accomplished in a fast, dynamic, and memory efficient way. Furthermore, the proposed approach only requires a portion of utterances from the old registrations during new registrations leading to an improved privacy preservation. Finally, the proposed approach provides an improved verification performance in the supervised mode.

Appendix A
Progressive Multi-Strided Random Buffer Sampling

In Pseudocode 1, $\text{num}_{\text{spk,utts}}(\cdot)$ computes the number of utterances per speaker $n_{\text{spk,utt}}$ by dividing the maximum allowed memory max_{mem} by the total number of speakers n_{tot}, finding the floor of the division, and converting the result to an integer using $\lfloor\cdot\rfloor$. The function $\text{collection}_{\text{indx}}(\cdot)$ computes the dictionary $\mathcal{C}_{\text{indx}}$ for the given key b and values of flattened list of speaker(s) per bucket. The function $\text{sample}_{\text{int-bkt}}(\cdot)$ computes the progressive features $\mathbf{Z}_{\text{buff}}^{\text{max}_{\text{mem}}}$ and the corresponding labels $\mathbf{y}_{\text{buff}}^{\text{max}_{\text{mem}}}$ with the maximum allowed memory size of max_{mem}.

Appendix B
Distance Metric

In the low-dimensional setting of latent features, $\mathbb{E}[\widetilde{\mathbf{z}}, \mathbf{z}]$, where $\widetilde{\mathbf{z}}$ and $\mathbf{z}$ denote the latent features of new and old registrations, respectively, inherits all the nice properties of

def $num_{\text{spk,utts}}\,(max_{\text{mem}}, \mathbf{n}_b, \mathbf{n}_b^{\text{reg}})$ **:**
  $n_{\text{tot}} = 0$
  for $n_b,\ n_b^{\text{reg}}$ **in** zip$(\mathbf{n}_b,\ \mathbf{n}_b^{\text{reg}})$ **:**
    $n_{\text{tot}}\ +=\ (n_b\ +\ n_b^{\text{reg}})$
    # Round down
    $n_{\text{spk,utt}}\ =\ \lfloor max_{\text{mem}}/n_{\text{tot}} \rfloor$
  return $n_{\text{spk,utt}}$
def $collection_{\text{indx}}\,(n_{s,\text{utt}}, n_{\text{spk,utt}}, \mathbf{n}_b, \mathbf{n}_b^{\text{reg}})$ **:**
  $\mathcal{C}_{\text{indx}}\ =\ \{\}$
  for $n_b,\ n_b^{\text{reg}}$ **in** zip$(\mathbf{n}_b,\ \mathbf{n}_b^{\text{reg}})$ **:**
    $\tilde{n}_b\ =\ n_b\ +\ n_b^{\text{reg}}$
    $\mathbf{indx}_{\text{utt}}^{(i)}\ =\ [i.n_{s,\text{utt}}, i.n_{s,\text{utt}} + n_{s,\text{utt}})$
    # $rand_{\text{sample}}(\mathbf{v}, n)$ randomly selects n
    # samples from $\mathbf{v}$
    # w/ or w/o replacement
    $\mathbf{indx}_{\text{utt},b}\ =\ [rand_{\text{sample}}\left(\mathbf{indx}_{\text{utt}}^{(i)}, n_{\text{spk,utt}}\right)$
    for i **in** range$(\tilde{n}_b)]$
    $\overline{\mathbf{indx}}_{\text{utt},b}\ =\ [u$ **for** u_s **in** $\mathbf{indx}_{\text{utt},b}$
    for u **in** $u_s]$
    $\mathcal{C}_{\text{indx}}[b] \leftarrow \overline{\mathbf{indx}}_{\text{utt},b}$
  return $\mathcal{C}_{\text{indx}}$
def $sample_{\text{int-bkt}}\,(\mathcal{C}_{\text{indx}}[b], \mathbf{zy}_b, \mathbf{zy}_{\text{init}}, perm = True)$ **:**
  if $perm$**:**
    $\mathcal{C}_{\text{indx}}[b] \leftarrow rand_{\text{sample}}\,(\mathcal{C}_{\text{indx}}[b], len(\mathcal{C}_{\text{indx}}[b]))$
  $\mathbf{z}_{\text{init}}, \mathbf{y}_{\text{init}}\ =\ \mathbf{zy}_{\text{init}}$
  $\mathbf{z}_b, \mathbf{y}_b\ =\ \mathbf{zy}_b$
  $\mathbf{z}_{\text{init}}.append(\mathbf{z}_b[\mathcal{C}_{\text{indx}}[b]])$
  $\mathbf{y}_{\text{init}}.append(\mathbf{y}_b[\mathcal{C}_{\text{indx}}[b]])$
  # Concatenate over batch dimension
  $\mathbf{Z}_{\text{buff}}^{max_{\text{mem}}} \leftarrow concat(\mathbf{z}_{\text{init}}, dim = 0)$
  $\mathbf{y}_{\text{buff}}^{max_{\text{mem}}} \leftarrow concat(\mathbf{y}_{\text{init}}, dim = 0)$
  return $\mathbf{Z}_{\text{buff}}^{max_{\text{mem}}}, \mathbf{y}_{\text{buff}}^{max_{\text{mem}}}$

Pseudocode 1. Functions for progressive multi-strided random buffer sampling.

def $opt\_spk\_bkt\,(\bar{\mathbf{z}}^{\text{eval}}, \mathbf{b}, \mathbf{s}_{\text{reg}}, n_{\text{round}})$ **:**
  # Previous $\mathbf{z}^{\text{eval}}$ and new $\widetilde{\mathbf{z}}^{\text{eval}}$ embedding
  # of registrations in evaluation mode
  $\mathbf{z}^{\text{eval}}, \widetilde{\mathbf{z}}^{\text{eval}}\ =\ \bar{\mathbf{z}}^{\text{eval}}$
  if $n_{\text{round}}\ ==\ 0$**:**
    $\mathbf{b}^*_{\text{sofar}}, \mathbf{s}^*_{\text{sofar}}\ =\ [], []$
  if $n_{\text{round}}\ >\ 0$**:**
    $\mathbf{b}^*_{\text{sofar}}.append(\mathbf{b}^*_{\text{reg}})$ from $n_{\text{round}} - 1$.
    $\mathbf{s}^*_{\text{sofar}}.append(\mathbf{s}^*_{\text{reg}})$ from $n_{\text{round}} - 1$.
  # Obtain prototypes for previously
  # registered speakers/buckets
  # for set of hold out utterances $\widetilde{\mathcal{P}}_{s,b}$
  # with cardinality of $|\widetilde{\mathcal{P}}_{s,b}|$
  $\boldsymbol{\mu}_{s,b} = \frac{1}{|\widetilde{\mathcal{P}}_{s,b}|} \sum_{i \in \widetilde{\mathcal{P}}_{s,b}} \mathbf{z}_{s,b}^{(i),\text{eval}}$
  for s_{reg} **in** $\mathbf{s}_{\text{reg}}$**:**
    for $\_, b$ **in** $enumerate(\mathbf{b})$**:**
      # Obtain average features
      # for new speakers/buckets
      $\widetilde{\boldsymbol{\mu}}_{s_{\text{reg}},b} = \frac{1}{|\widetilde{\mathcal{P}}_{s_{\text{reg}},b}|} \sum_{i \in \widetilde{\mathcal{P}}_{s_{\text{reg}},b}} \widetilde{\mathbf{z}}_{s_{\text{reg}},b}^{(i),\text{eval}}$
      # Obtain L2 pairwise distance
      $d(\widetilde{\boldsymbol{\mu}}_{s_{\text{reg}},b}, \boldsymbol{\mu}_{s,b}) = \|\widetilde{\boldsymbol{\mu}}_{s_{\text{reg}},b} - \boldsymbol{\mu}_{s,b}\|^2$
    # Form optimal buckets $\mathbf{b}_{\text{reg}}$
    # and corresponding speakers $\mathbf{s}_{\text{reg}}$
    $\mathbf{s}_{\text{reg}}, \mathbf{b}_{\text{reg}}\ \leftarrow$
    $s_{\text{reg}}, b_{\text{reg}} = \text{argmin}_{s,b}\, d(\widetilde{\boldsymbol{\mu}}_{s_{\text{reg}},b}, \boldsymbol{\mu}_{s,b})$
  Obtain $\mathbf{b}^*_{\text{reg}}$ & $\mathbf{s}^*_{\text{reg}}$ *by Algorithm. 4.*
  return $\mathbf{b}^*_{\text{reg}}, \mathbf{s}^*_{\text{reg}}, \mathbf{b}^*_{\text{sofar}}, \mathbf{s}^*_{\text{sofar}}$

Pseudocode 2. Compute optimal new speakers/buckets.

generalized energy distance and maximum mean discrepancy (MMD) [25]. The term $\mathbb{E}[\widetilde{\mathbf{z}}, \mathbf{z}]$ can be written as follows:

$$\mathbb{E}[\widetilde{\mathbf{z}}, \mathbf{z}]^2 = \text{tr}\left(\widetilde{\boldsymbol{\Sigma}}\right) + \text{tr}(\boldsymbol{\Sigma}) + \|\widetilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2 \tag{11}$$

where $\text{tr}(\boldsymbol{\Sigma})$ and $\boldsymbol{\mu}$ denote the trace of covariance matrix $\boldsymbol{\Sigma}$ and mean vector for $\mathbf{z}$, and the corresponding terms for $\widetilde{\mathbf{z}}$ are similarly defined. The first two terms in (11) can be upper bounded by constants. This is due to the boundness of the contrastive features in the latent space. Consequently, the minimization of (11), to obtain the optimal buckets and corresponding speakers, reduces to the minimization of the last term. In other words, the shortest $L2$ pairwise distance is used as the metric to find the optimal buckets.

On the other hand, the minimax rate optimal estimator of MMD with the rate of $m^{-0.5} + n^{-0.5}$ can be written as follows [26]:

$$\text{MMD}_{n,m} = \left\|\widetilde{\boldsymbol{\mu}}_{P_n} - \boldsymbol{\mu}_{Q_m}\right\|_{\mathcal{H}} \tag{12}$$

where $\widetilde{\boldsymbol{\mu}}_{P_n} := (1/n)\Sigma_{i=1}^n k(.,\widetilde{\mathbf{X}}_i)$ in which $\widetilde{\mathbf{X}}_i \overset{\text{i.i.d.}}{\sim} P$ and $\boldsymbol{\mu}_{Q_m} := (1/m)\Sigma_{i=1}^m k(.,\mathbf{X}_i)$ in which $\mathbf{X}_i \overset{\text{i.i.d.}}{\sim} Q$ for a continuous positive definite real-valued kernel $k(.,.)$ and the corresponding reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. It is observed that (12) provides a similar result in terms of selecting the optimal bucket(s) in the low dimensional setting. A similar argument holds for the U-statistic variant of $\text{MMD}_{n,m}$.

Appendix C
Computing Optimal New Speakers/Buckets for Dynamic Registration

In Pseudocode 2, opt_spk_bkt(·) provides a method to compute $\mathbf{b}^*_{\text{reg}}$, $\mathbf{s}^*_{\text{reg}}$, $\mathbf{b}^*_{\text{sofar}}$, and $\mathbf{s}^*_{\text{sofar}}$. Different steps of the aforementioned function are commented in Pseudocode 2. To obtain the longest unique sequence of the optimal buckets for registering new speakers in each round, dynamic programming Algorithm 4 of decision type is used. It receives the

Algorithm 4 Longest Unique Sequence of Optimal Buckets per Registration

Input: The sequence of optimal buckets for new speaker registrations $\mathbf{b}_{\text{reg}}$ and corresponding speakers $\mathbf{s}_{\text{reg}}$.
Output: The sequence of longest optimal unique per registration buckets $\mathbf{b}^*_{\text{reg}}$ and the corresponding index of new registered speakers $\mathbf{s}^*_{\text{reg}}$.
Subproblem: The sequence of longest optimal unique per registration buckets $\mathbf{b}^*_{\text{reg}}[:i]$ for the set of new speakers in the interval $[N, i)$ for $i \in \{N, \ldots, N + len(\mathbf{s}_{\text{reg}}) - 1\}$.
Relation: Recursive computations to obtain the sequence of longest optimal unique per registration buckets in (13).
Topological Order: Sub-problem $\mathbf{b}^*_{\text{reg}}[:i+1]$ only depends on strictly smaller i, so it is acyclic, i.e., increase i for $i = N, \ldots, N + len(\mathbf{s}_{\text{reg}}) - 1$.
Base Case: The empty set is always achieved for $\mathbf{b}^*_{\text{reg}}[:N] = \emptyset$.
Original Problem: The sequence of longest optimal unique per registration buckets for the entire set of new speakers, i.e., $\mathbf{b}^*_{\text{reg}}[:N + len(\mathbf{s}_{\text{reg}})]$.

full list of optimal buckets and corresponding speakers for the current round as the inputs. Then, the longest unique sequence of the optimal buckets is achieved according to the recursive call as follows:

$$\mathbf{b}^*_{\text{reg}}[:i+1] = \begin{cases} \mathbf{b}^*_{\text{reg}}[:i], & b^*_i \in \mathbf{b}^*_{\text{reg}}[:i] \\ \mathbf{b}^*_{\text{reg}}[:i] \cup \{b^*_i\}, & b^*_i \notin \mathbf{b}^*_{\text{reg}}[:i]. \end{cases} \tag{13}$$

In (13), the optimal bucket at index i, b^*_i, is added only if it does not already exist in the set of new speakers in the interval $[N, i)$. The standard steps for the dynamic process to find the solution for the subset of the original problem using the subproblem for the base case and the relation in (13) are described in Algorithm 4. Consequently, by increasing the index i, the entire list of optimal buckets is covered starting from the base case in the bottom-up way. The proposed dynamic programming algorithm only requires the linear time complexity of $O(N_{\text{reg}})$ for the worst case, i.e., $\text{len}(\mathbf{s}_{\text{reg}}) = N_{\text{reg}}$. This is due to the fact that the sequence of new speaker registrations is progressively reduced after each round.

Appendix D
Computing Properties for Registering

In Pseudocode 3, prop_reg(·) provides the required properties for dynamic registrations of the new speakers. For each bucket b, there exist four different patterns/strategies provided by the function $\text{strg}_{\text{reg}}(\cdot)$. The function $\text{strg}_{\text{reg}}(\cdot)$ first unifies the specific arguments for different strategies through the partial(·) operation, similar to the partial(·) in Python. The aforementioned patterns are used as the keys for the strategy dictionary $\widetilde{\text{strg}}$ with the values representing different logics for registration. Consequently, the appropriate pattern is selected and returned if the corresponding logic is fulfilled.

```
def strg_1 (s*_reg, b*_reg, b, n_b) :
    return s*_reg[find(b ∈ b*_reg)], pattern_1, n_b, 1
def strg_2 (s*_sofar, b*_sofar, b, n_b) :
    return (s*_sofar[find(b ∈ b*_sofar)], pattern_2,
                n_b + len(s*_sofar[find(b ∈ b*_sofar)]), 0)
def strg_3 (s*_reg, b*_reg, s*_sofar, b*_sofar, b, n_b) :
    return (s*_reg[find(b ∈ b*_reg)]
              ∪ s*_sofar[find(b ∈ b*_sofar)], pattern_3,
                n_b + len(s*_sofar[find(b ∈ b*_sofar)]), 1)
def ~strg_4 (n_b) :
    return [], pattern_4, n_b, 0
def strg_reg (b, b*_reg, b*_sofar, s*_reg, s*_sofar) :
    # Unify args using partial(.)
    # operation in Python
    ~strg_1 = partial(strg_1, s*_reg, b*_reg, b)
    ~strg_2 = partial(strg_2, s*_sofar, b*_sofar, b)
    ~strg_3 = partial(strg_3, s*_reg, b*_reg, s*_sofar, b*_sofar, b)
    ~strg = {~strg_1: (b ∈ b*_reg & b ∉ b*_sofar),
             ~strg_2: (b ∉ b*_reg & b ∈ b*_sofar),
             ~strg_3: (b ∈ b*_reg & b ∈ b*_sofar),
             ~strg_4: (b ∉ b*_reg & b ∉ b*_sofar)}
    for ~strg_selct, logic in ~strg.items():
        if logic:
            return ~strg_selct
def prop_reg (b, n_b, b*_reg, b*_sofar, s*_reg, s*_sofar) :
    ñ_b, ñ^reg_b, S_reg, P_reg = [], [], {}, {}
    for b, n_b in zip(b, n_b):
        ~strg_selct = strg_reg (b, b*_reg, b*_sofar, s*_reg, s*_sofar)
        S_reg[b], P_reg[b], ñ_b[b], ñ^reg_b[b] ← ~strg_selct (n_b)
    return ñ_b, ñ^reg_b, S_reg, P_reg
```

Pseudocode 3. Compute properties for registering.

The first strategy $\text{strg}_1(\cdot)$ is selected, if b belongs to $\mathbf{b}^*_{\text{reg}}$ and not to $\mathbf{b}^*_{\text{sofar}}$. The set of indices of new speakers in which $b \in \mathbf{b}^*_{\text{reg}}$ is obtained using find(·) operation as the first term to return. The corresponding pattern status of pattern_1 is returned as the second term; the number of speakers per bucket in this case n_b is returned as the third term, and finally, the number of new speaker to be registered in this bucket under this strategy is one. The third strategy $\text{strg}_3(\cdot)$ is selected, if b belongs to both $\mathbf{b}^*_{\text{sofar}}$ and $\mathbf{b}^*_{\text{reg}}$. The set of indices of new speakers in which $b \in \mathbf{b}^*_{\text{reg}}$ and $b \in \mathbf{b}^*_{\text{sofar}}$ is obtained using find(·) operation as the first term to return. The corresponding pattern status of pattern_3 is selected; the number of speakers per bucket in this case $n_b + \text{len}(\mathbf{s}^*_{\text{sofar}}[\text{find}(b \in \mathbf{b}^*_{\text{sofar}})])$ is returned, and finally, the number of new speaker to be registered in this bucket under this strategy is one. Similarly, the rest of strategies are selected based on the corresponding logic.

Appendix E
Computing Properties for Removing

In Pseudocode 4, prop_unreg(·) provides the required properties for removing the given set of speakers from the pool

```
def strg_1 (b, b_unreg, s_res) :
    return (s_res[find(b ∈ b_unreg)], pattern_1,
                                  len(s_res[find(b ∈ b_unreg)]))
def strg_2 (n_b, b, b_unreg, s_res) :
    return s_res[find(b ∉ b_unreg)], pattern_2, n_b
def strg_unreg (b, n_b, b_unreg) :
        strg̃_2 = partial(strg_2, n_b)
        strg̃ = {strg̃_1 : b ∈ b_unreg,
                         strg̃_2 : b ∉ b_unreg}
    for strg̃_selct, logic in strg̃.items() :
        if logic:
            return strg̃_selct
def prop_unreg (b, n_b, b_unreg, s_res) :
    ñ_b, S_res, P_unreg = [], {}, {}
    for b, n_b in zip(b, n_b):
        strg̃_selct = strg_unreg (b, n_b, b_unreg)
        S_res[b], P_unreg[b], ñ_b[b] ←
                         strg̃_selct (b, b_unreg, s_res)
    return ñ_b, S_res, P_unreg
```

Pseudocode 4. Compute properties for removing.

of already registered speakers. There exist two different patterns/strategies provided by the function $\text{strg}_{\text{unreg}}(\cdot)$. The function $\text{strg}_{\text{unreg}}(\cdot)$ first unifies the specific arguments for different strategies through the partial$(\cdot)$ operation. The aforementioned strategies are used as the keys for the strategy dictionary $\widetilde{\text{strg}}$ with the values representing different logics for removal. Consequently, the appropriate pattern is selected and returned if the corresponding logic is fulfilled.

The first strategy is selected, if b belongs to $\mathbf{b}_{\text{unreg}}$. The set of indices of residual speakers in which $b \in \mathbf{b}_{\text{unreg}}$ is obtained using find$(\cdot)$ operation as the first term to return. The corresponding pattern status of pattern_1 is returned as the second term, and the number of remaining speakers per bucket $\tilde{\mathbf{n}}_b[b]$, in this case len($\mathbf{s}_{\text{res}}$[find($b \in \mathbf{b}_{\text{unreg}}$)]), is returned as the last term. Similarly, the desired properties are returned if the condition for the first strategy is not fulfilled. In this case, $\tilde{\mathbf{n}}_b[b]$ is set to the initial state before removal, i.e., n_b.

REFERENCES

[1] E. Rubio-Drosdov, D. Díaz-Sánchez, F. Almenárez, P. Arias-Cabarcos, and A. Marín, "Seamless human-device interaction in the Internet of Things," *IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 490–498, Nov. 2017.

[2] P. Cheng, I. E. Bagci, J. Yan, and U. Roedig, "Towards reactive acoustic jamming for personal voice assistants," in *Proc. 2nd Int. Workshop Multimedia Privacy Secur.*, Jan. 2018, pp. 12–17.

[3] P. Cheng, I. E. Bagci, J. Yan, and U. Roedig, "Smart speaker privacy control–acoustic tagging for personal voice assistants," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 144–149.

[4] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2017, pp. 4080–4090.

[5] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2016, pp. 3637–3645.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 1126–1135.

[7] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias, "Brain-inspired replay for continual learning with artificial neural networks," *Nature Commun.*, vol. 11, no. 1, pp. 1–14, Aug. 2020.

[8] J. von Oswald, C. Henning, B. F. Grewe, and J. Sacramento, "Continual learning with hypernetworks," 2019, *arXiv:1906.00695*.

[9] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 3584–3594.

[10] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 4879–4883.

[11] P. Khosla et al., "Supervised contrastive learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 18661–18673.

[12] M. Sang, H. Li, F. Liu, A. O. Arnold, and L. Wan, "Self-supervised speaker verification with simple Siamese network and self-supervised regularization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 6127–6131.

[13] H. Zhang, Y. Zou, and H. Wang, "Contrastive self-supervised learning for text-independent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 6713–6717.

[14] W. Xia, C. Zhang, C. Weng, M. Yu, and D. Yu, "Self-supervised text-independent speaker verification using prototypical momentum contrastive learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 6723–6727.

[15] A. Sholokhov, X. Liu, M. Sahidullah, and T. Kinnunen, "Baselines and protocols for household speaker recognition," in *Proc. Speaker Lang. Recognit. Workshop*, Jun. 2022, pp. 185–192.

[16] O. Sadjadi, C. Greenberg, E. Singer, L. Mason, and D. Reynolds. *NIST 2021 Speaker Recognition Evaluation Plan*. Dec. 7, 2021. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=932697

[17] E. D. Demaine, J. Iacono, and S. Langerman, "Retroactive data structures," *ACM Trans. Algorithms*, vol. 3, no. 2, p. 13, May 2007.

[18] A. Shahmansoori. (Sep. 2023). *dynamic-consent-management*. [Online]. Available: https://github.com/arash-shahmansoori/dynamic-consent-management.git

[19] 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.

[20] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," 2017, *arXiv:1706.08612*.

[21] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science, vol. 11217, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Sep. 2018, pp. 3–19.

[22] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[23] H. Liu, Z. Li, D. Hall, P. Liang, and T. Ma, "Sophia: A scalable stochastic second-order optimizer for language model pre-training," 2023, *arXiv:2305.14342*.

[24] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[25] S. Chakraborty and X. Zhang, "A new framework for distance and kernel-based metrics in high dimensions," *Electron. J. Statist.*, vol. 15, no. 2, pp. 5455–5522, Jan. 2021, doi: 10.1214/21-EJS1889.

[26] I. O. Tolstikhin, B. K. Sriperumbudur, and B. Schölkopf, "Minimax estimation of maximum mean discrepancy with radial kernels," in *Advances in Neural Information Processing Systems*, vol. 29, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/5055cbf43fac3f7e2336b27310f0b9ef-Paper.pdf

Arash Shahmansoori received the B.Sc. degree (Hons.) in bioelectric engineering from Tehran Polytechnic, Tehran, Iran, in 2010, the M.Sc. degree in signal processing from the Norwegian University of Science and Technology, Trondheim, Norway, in 2012, and the Ph.D. degree in telecommunications engineering from the Universitat Autònoma de Barcelona, Barcelona, Spain, in 2017.

He worked as a Post-Doctoral Researcher at the Institute of Electronics and Telecommunications of Rennes, University of Rennes 1, Rennes, France, in 2018. He was a Researcher at the Mitsubishi Electric Research and Development Centre Europe, Rennes, in 2019. He has been a Senior Post-Doctoral Researcher at the School of Computer Science and Information Technology, University College Cork, Cork, Ireland, since 2020. He has published several papers in prestigious journals, conferences, and patents in the fields of wireless communications, signal processing, and machine learning. His current research interests include generative artificial intelligence (AI), autonomous agents, and cognitive systems.

Utz Roedig (Member, IEEE) received the Dr.-Ing. degree in computer science from the Darmstadt University of Technology, Germany, Darmstadt, in 2002.

He was a Professor at Lancaster University, Lancaster, U.K., where he led the Academic Centre of Excellence in Cyber Security Research. He has also held research positions at the Darmstadt University of Technology and the University College Cork (UCC), Cork, Ireland. He has been the Head of the School of Computer Science and Information Technology, UCC, since September 2021. He is a Co-Principal Investigator of the CONNECT Centre for Future Networks, Dublin, Ireland, and a Principal Investigator of the Personal Voice Assistant Security and Privacy Project. He has published over 160 peer-reviewed articles and several patents in this field. His research interests are computer networks and security, especially for the Internet of Things (IoT).

Dr. Roedig frequently serves as a Technical Program Committee (TPC) Member for international conferences, such as Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), Embedded Wireless Systems and Networks (EWSN), Information Processing in Sensor Networks (IPSN). He has received research grants from Europe, Science Foundation Ireland (SFI), Engineering and Physical Sciences Research Council (EPSRC), and industry. He is a Grant Reviewer for international funding bodies, such as EPSRC, U.K.; European Social Fund (ESF), Europe; and Fonds Voor Wetenschappelijk Onderzoek-Vlaanderen (FWO), Belgium.