INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Ordered and Binary Speaker Embedding

Jiaying Wang[1,3], *Xianglong Wang*[3], *Namin Wang*[2], *Lantian Li*[3*], *Dong Wang*[1*]

[1]Center for Speech and Language Technologies, BNRist, Tsinghua University, China
[2]Huawei Cloud, China [3]Beijing University of Posts and Telecommunications, China
*Corresponding authors: lilt@bupt.edu.cn, wangdong99@mails.tsinghua.edu.cn

Abstract

Modern speaker recognition systems represent utterances by embedding vectors. Conventional embedding vectors are dense and non-structural. In this paper, we propose an ordered binary embedding approach that sorts the dimensions of the embedding vector via a nested dropout and converts the sorted vectors to binary codes via Bernoulli sampling. The resultant ordered binary codes offer some important merits such as hierarchical clustering, reduced memory usage, and fast retrieval. These merits were empirically verified by comprehensive experiments on a speaker identification task with the VoxCeleb and CN-Celeb datasets.

Index Terms: speaker recognition, ordered binary embedding, auto-encoder

1. Introduction

Speaker recognition is the process of recognizing the identity of a person from his/her voice. Due to its traits of being user-friendly, non-intrusive, non-touching, and low privacy, speaker recognition has found broad real-life applications [1]. Modern speaker recognition systems are mostly based on the concept of ‘embedding’, i.e., represent a variable-length utterance by a fixed-dim dense vector. Traditional speaker embedding is based on statistical models, in particular the i-vector model [2]. Recently, deep neural nets become the most popular embedding models [3, 4, 5, 6] and achieved state-of-the-art performance in various benchmarks [7, 8, 9, 10]. Among all the embedding models, the x-vector model achieved the most success [5].

Despite the broad success of deep speaker embedding, the dense embedding vectors are not suitable for large-scale identification tasks. For instance, per our experiment, identifying a person from 1,251 candidates using 32-dim x-vectors costs 50 ms on a 1.2 GHz CPU and with the highly optimized Scipy package. This amounts to 15.5 hours of CPU time per query if the size of candidates is 1.4 billion, the population of China. This is unacceptable for most real applications. To respond to this challenge, the recent CNSRC evaluation has set a large-scale identification task and required the participants to report the time cost for search [11].

A known approach to accelerating the search speed is by hierarchical clustering [12, 13]. By this approach, enrolled speakers are clustered according to their similarity and multiple-level clustering forms a decision tree. This approach largely reduces the search time, though the depth of the tree must be carefully controlled to avoid a substantial performance drop. Another direction is to use binary codes [14, 15]. This approach converts dense embeddings to binary codes, and the similarity is computed as the Hamming distance which is much faster than computing the cosine distance. The shortage of this approach is that the binarization process may lose precision, and the search still needs to scan all the candidates.

In this paper, we propose a novel approach that involves the advantage of both hierarchical clustering and binary codes. We design an ordered binary auto-encoder (obAE) model whose encoder converts a dense embedding to an ordered binary (OB) code, where the bits of the code are ordered according to their importance for the decoder to recover the input dense embedding. The architecture is shown in Figure 1(a), and the main architecture involves a nested dropout [16] and a Bernoulli sampling. Once the model is trained, the encoder can be used to produce OB codes, as shown in Figure 1(b)(c). The implementation is as simple as several lines of Python code, and the additional computation cost is negligible. More details will be presented in Section 3.2.

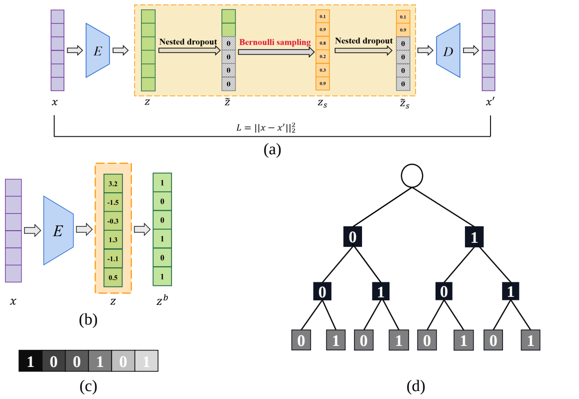


Figure 1: *(a) The obAE model in the training phase. Details are in Section 3.2. (b) The obAE model in the inference phase. The dense vector x is converted to logits z by the encoder, which is then converted to an OB code z^b. (c) An OB code example, with the grey level indicating the importance; (d) Binary tree formed by OB codes. The top circle is the starting point of the search.*

On the one hand, OB codes are efficient in both search speed and memory usage, the same merit shared by other binary codes such as those derived by local sensitive Hashing (LSH) [17]. However, since the OB code here is ‘learned’ to minimize the reconstruction error, it is more task-oriented, under the assumption that the reconstruction error is consistent with the similarity used in the search task, which is the case for speaker recognition as we will show in the experiments. On the other hand, since the bits are ordered in OB codes, it natu-

This work was supported by the National Natural Science Foundation of China under Grant No.62171250, and also the Huawei Cloud Research Program under the project No.TC20220615035.

10.21437/Interspeech.2023-1565

rally forms a binary tree that hierarchically clusters the enrolled speakers, as shown in Figure 1(d). This offers a fast retrieval by a top-down tree traversal, a merit shared with the conventional hierarchical clustering approach [13]. An advantage of the binary tree, however, is that the tree structure is optimized jointly with the code, leading to a better accuracy-efficiency trade-off. Moreover, searching over the binary tree is just a sequence of bit matching, which is much faster than the conventional tree search where at each node the query embedding needs to be compared with all the cluster centroids.

To verify the advantage of the OB, we conducted comprehensive experiments on speaker identification (SID) tasks with the VoxCeleb [18] and CN-Celeb [19] datasets. To our best knowledge, this paper is the first attempt to investigate ordered and binary speaker embedding.

2. Related work

Making features ordered is a long-standing topic. Principal components analysis (PCA) is perhaps the most popular approach [20]. Since the emergence of deep models, multiple research has been conducted to impose dimensional orderliness in the latent space of neural nets, especially auto-encoders (AEs). Rippel et al. [16] proposed a nested dropout algorithm. That applies structured masks on the latent features to encourage leading dimensions to take more important role. Ladjal et al. [21] proposed a PCA-like AE. They designed an iterative training strategy that progressively increases the latent dimension, so that leading dimensions are learned first and more sufficiently. The merits of ordered features have been in multiple fields such as image compression, retrieval, and generation [22, 23, 24, 25]. As far as we know, there is no investigation on ordered speaker embedding.

For binary speaker code, [14] presented a binarization approach based on kernel LSH within the GMM-UBM framework. Li et al. [15] presented an approach based on LSH as well, but performed the test on i-vectors. Both studies reported reduced storage/memory and improved search speed with little or no loss of accuracy.

For ordered binary codes, Rippel et al. [16] reported a simple 'thresholding' approach, that cats a dense vector to binary codes according to a predefined threshold. Xu et al. [22] presented a discretization approach based on K-means clustering as in VQ-VAE [26]. Our obAE model differs from the above studies in that we obtain binary codes with Bernoulli sampling, and train the model with the reparametrization trick [27]. We found this approach is simple in implementation and the training is stable.

3. Method

3.1. Ordered AE (oAE)

By setting a reconstruction loss and an appropriate information bottleneck [28, 29], AE can discover a low-dimensional latent space that retains the most important information for describing the data. Compared to PCA, AE is a non-linear model and is more flexible. The cost of the flexibility, however, is that the latent space loses the property of dimensional orderliness. Nested dropout on the latent features [16, 22] can recover the order, leading to an ordered AE (oAE). Note that oAE has been proposed in previous studies [16, 22], though not investigated in speaker recognition.

The oAE architecture and nested dropout are illustrated in Figure 2. Let z denote the latent feature of an input data x. Nested dropout samples a dimension index i following a uniform distribution, and then masks all the dimensions of z after the i-th dim, i.e., set them to zeros. The masked feature, denoted by $\tilde{z} = [z_{\leq i}, 0, .., 0]$, is then used to produce x', the reconstruction of the input x. The model is trained with the MSE loss, as usual AEs.

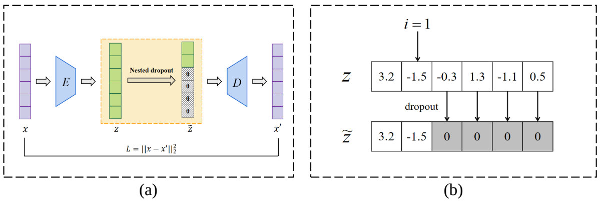


Figure 2: *Illustration of (a) oAE architecture and (b) nested dropout, where z and $\tilde{z}$ represent the latent features before and after dropout. Note that* ***all*** *the dimensions after the i-th dim are dropped out, i.e., set to 0. This is the main difference between nested dropout and conventional unit dropout.*

Theoretical analysis [16, 22] shows that the *incremental* value of the mutual information between x and $z_{\leq i}$, i.e., $\mathrm{MI}(x, z_{\leq i}) - \mathrm{MI}(x, z_{\leq i-1})$, declines when i increases. This means that z_i is ordered according to the information it contains regarding the input, hence the importance for its reconstruction. Another theoretical result is that oAE recovers PCA when the encoder and decoder are linear and share parameters [16]. This establishes a more close link between PCA and AE models.

3.2. Ordered binary AE (obAE)

3.2.1. Architecture

We extend ordered AE to ordered binary AE by introducing a Bernoulli sampling. The model architecture has been shown in Figure 1(a). Briefly, we treat the masked feature $\tilde{z}$ as the parameter of a Bernoulli distribution and sample a binary code z_s from it. This is formally written by $z_s \sim \mathrm{Bernoulli}(p)$ where $p = \sigma(\tilde{z})$, with $\sigma(\cdot)$ the sigmoid function. The sampled z_s is then masked following the same way as $\tilde{z}$, producing a masked binary code $\tilde{z}_s$. Finally, $\tilde{z}_s$ passes the decoder and the reconstructed input x' is produced. Again, the training is based on MSE loss.

In the test phase, both the masking and the Bernoulli sampling are not required, and we simply use the following hash functions to obtain the OB code z^b:

$$z_i^b = \begin{cases} 1 & z_i \geq 0 \\ 0 & z_i < 0 \end{cases} \tag{1}$$

3.2.2. Relaxed Bernoulli sampling

A difficulty when training the obAE model is that the gradient cannot be back-propagated to the encoder as z impacts the output via a Bernoulli sampler. We solve the problem by using the resampling trick as in VAE [27]. Specifically, we first sample u from a uniform distribution on [0,1], and then produce z_s as follows:

$$z_{si} = \begin{cases} 1 & u_i \leq p_i \\ 0 & u_i > p_i \end{cases} \tag{2}$$

where $p = \sigma(\tilde{z})$.

It is easy to verify that z_s generated in this way follows $\mathrm{Bernoulli}(p)$, and $\tilde{z}$ can receive gradients as it is not related to the sampler. However, Eq. (2) is a step function so cannot pass gradient. A sigmoid function can be used as a relaxed version, leading to the following procedure:

$$z_s = \sigma \left(\frac{\log(\frac{u}{1-u}) + \log(\frac{p}{1-p})}{T} \right), \tag{3}$$

where T controls the sharpness of the distribution of z_s. The smaller the value T, the higher the probability that z_s concentrates around 0 or 1. In the limit case $T \rightarrow 0$, the distribution of z_s approaches Bernoulli(p). In our experiments, we set T to 0.1, and implement this sampling by the *RelaxedBernoulli*$(\cdot)$ function supported by PyTorch[1].

4. Experiments

In this section, we test the OB codes on speaker identification tasks with VoxCeleb [18] and CN-Celeb [19] datasets. We first present the data and setups, and then report the results with ordered dense features and ordered binary codes in sequence.

4.1. Data

Two datasets were used in our experiments: VoxCeleb [18] and CN-Celeb [19], and their information is presented in Table 1. It is worth noting that for VoxCeleb1, we selected the longest three utterances of each speaker for enrollment and the remaining for test. For CN-Celeb.E, to avoid possible annotation errors in short utterances, we removed the test utterances whose duration is less than 2 seconds.

Table 1: *Data description*

VoxCeleb	Train VoxCeleb2.dev	Enroll VoxCeleb1	Test VoxCeleb1
# of Spks # of Utters	5,994 1,092,009	1,251 3,753	1,251 149,763
CN-Celeb	Train CN-Celeb.T	Enroll CN-Celeb.E	Test CN-Celeb.E
# of Spks # of Utters	2,793 632,740	196 196	196 14,124

4.2. Settings

We use two publicly available x-vector models released in the Sunine toolkit[2]: one was trained on VoxCeleb2.dev, and the other was trained on CN-Celeb.T. The structure of both models is ResNet34 [30] with squeeze-and-excitation (SE) layers [31], and the dimensionality of the x-vectors is 256. We use the VoxCeleb model and the CN-Celeb model to extract x-vectors of the utterances in VoxCeleb and CN-Celeb, respectively.

For oAE and obAE models, the encoder and the decoder are both linear, though their parameters are not shared. We choose this simple structure for a direct comparison with PCA. The source code is available online[3] to help readers reproduce our experiments.

4.3. Baseline

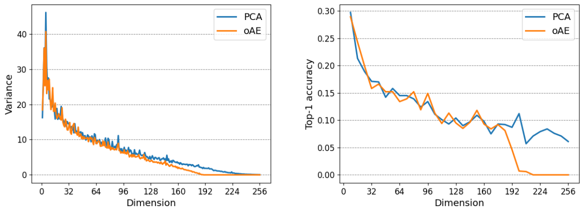
Firstly, the Top-k accuracy with the original x-vector (256 dims, or equally 8,192 bits) is reported in Table 2. It can be observed that although the number of speakers in CN-Celeb.E is much smaller than that in VoxCeleb1, the Top-k accuracies on CN-Celeb.E are clearly inferior to those on VoxCeleb1, indicating that CN-Celeb.E is a more challenging dataset for SID.

Table 2: *Top-k accuracy with the original x-vector.*

	VoxCeleb1	**CN-Celeb.E**
Top-1	0.959	0.706
Top-3	0.984	0.800
Top-5	0.989	0.844

4.4. Orderliness Test

In the second experiment, we test the orderliness of the features produced by oAE and PCA. Figure 3 illustrates (a) the variance of each dimension and (b) the Top-1 accuracy with partial dimensions. In the partial-dimension test, each test involves 8 consecutive dimensions, resulting in 32 data points on each curve in Figure 3(b).



(a) *Variance test* (b) *Performance test*

Figure 3: *Orderliness test with PCA and oAE, using x-vectors in VoxCeleb1.*

From Figure 3(a), one can observe that the variance of each dimension is quite similar with oAE and PCA, and the leading dimensions show larger variations. This demonstrates that oAE learns ordered representations as PCA. However, oAE is not a full copy of PCA. In particular, oAE performs even better than PCA on tail dimensions. The difference should be attributed to the fact that oAE does not enforce parameter sharing between the encoder and decoder while PCA does.

From Figure 3(b), we can see that the partial-dimension performances of PCA and oAE are also similar and the trends are the same: the leading dimensions are more important than the tail dimensions in the SID task. This double confirms that the oAE model can learn ordered representations. Interestingly, this learning is based on reconstruction error but works well on the SID task, suggesting some potential link between the reconstruction error and the cosine similarity used in SID.

4.5. Binary Test

In the third experiment, we test the OB codes derived from the obAE model. In this experiment, the dimensionality of the latent representation is 256, therefore the derived binary code involves 256 bits in total. We will select the first $i \leq 256$ bits to perform the test.

For a better comparison, we choose two alternative binary codes as references: one is produced from the original x-vector with local sensitive Hashing (LSH) [32, 15], denoted by LSH. And the other one is produced from the PCA-transformed x-vector with LSH, denoted by PCA-LSH. Results on the VoxCeleb1 and CN-Celeb.E datasets are reported in Table 3 and Table 4, respectively.

It can be seen that PCA-LSH and obAE significantly outperform LSH when the code size is small. However, as the code size becomes larger, this advantage will be gradually reduced and finally, the performance of the three codes tends to be the same. This is not surprising as LSH can perfectly represent the dense vector if sufficient Hash functions are used, according to the random matrix theory [32]. This means that the advantage

[1] https://pytorch.org/docs/stable/distributions.html#relaxedbernoulli
[2] https://gitlab.com/csltstu/sunine
[3] https://github.com/AlexGranger-scn/OAE

Table 3: *Top-k Acc with three binary codes on VoxCeleb1.*

Bits		20	40	80	120	160	256
LSH	Top-1	0.094	0.273	0.543	0.684	0.759	0.847
	Top-3	0.176	0.412	0.689	0.808	0.865	0.924
	Top-5	0.227	0.481	0.747	0.851	0.898	0.945
PCA-LSH	Top-1	0.126	0.350	0.599	0.709	0.768	0.847
	Top-3	0.233	0.514	0.746	0.830	0.872	0.924
	Top-5	0.297	0.588	0.799	0.870	0.904	0.945
obAE	Top-1	0.182	0.440	0.681	0.779	0.813	0.824
	Top-3	0.316	0.616	0.822	0.890	0.911	0.913
	Top-5	0.392	0.690	0.870	0.923	0.939	0.939

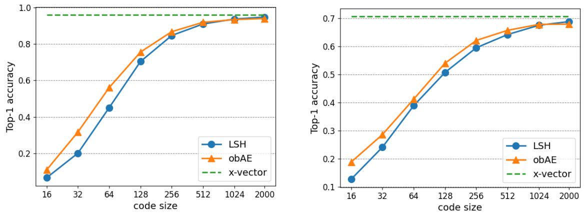
Table 4: *Top-k Acc with three binary codes on CN-Celeb.E.*

Bits		20	40	80	120	160	256
LSH	Top-1	0.157	0.293	0.432	0.502	0.543	0.595
	Top-3	0.266	0.410	0.546	0.612	0.653	0.703
	Top-5	0.326	0.471	0.602	0.664	0.702	0.750
PCA-LSH	Top-1	0.183	0.329	0.462	0.519	0.544	0.595
	Top-3	0.323	0.478	0.588	0.634	0.655	0.701
	Top-5	0.403	0.550	0.647	0.687	0.706	0.748
obAE	Top-1	0.197	0.353	0.495	0.551	0.579	0.588
	Top-3	0.357	0.518	0.639	0.688	0.708	0.719
	Top-5	0.446	0.594	0.705	0.743	0.762	0.774

any binary code may achieve is in the regime of limited code capacity. And just in this regime, obAE shows a clear advantage over LSH, no matter whether PCA is employed.

4.6. Bit Test

In the fourth experiment, we make a 'bit test' to find out how many bits of the LSH and obAE codes can obtain comparable performance to the original x-vector. To achieve this goal, we trained an obAE model with the dimensionality of the latent space set to 2,000, resulting in binary codes of 2,000 bits. Once trained, the first $i \leq$ 2,000 bits are selected to make the performance test, as shown in Figure 4.



(a) *VoxCeleb1*

(b) *CN-Celeb.E*

Figure 4: *Bit test with LSH and obAE codes.*

It can be seen that as the code size increases, the Top-1 accuracy of the obAE codes gradually improves and eventually converges. Compared to the original x-vector (8,192 bit) in Table 2, it seems that the OB codes could obtain a competitive performance when the code size reaches 1,000. This indicates that there are lots of redundancy in the original x-vector, and binary codes may represent the full information with less storage.

Finally, we see the performance of the obAE codes is consistently superior to the LSH codes, but ultimately they converge together. Again, this indicates that LSH works well with sufficient code size, but obAE can achieve better performance with limited code capacity. This feature is clearly attributed to the orderliness of the OB codes.

4.7. Speed Test

As mentioned in Section 1, the ordered and binary properties of the OB codes can be applied to gain fast retrieval. Specifically, the OB codes form a binary tree, where the hierarchy is just the dimension index, as shown in Figure 5. If we assume that each leaf node of the tree corresponds to a single speaker, then for any query OB code, a simple top-down traversal through the tree will reach the matched speaker, leading to very fast retrieval. Most importantly, the complexity of this traversal-based retrieval is independent of the size of the speakers. This means it is particularly suitable for search in a large population.

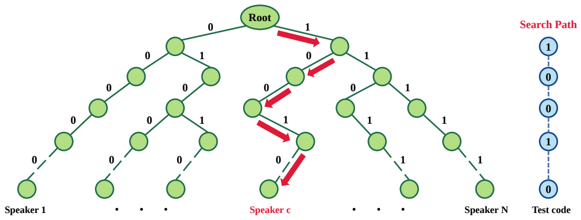


Figure 5: *An illustration of the binary search tree. The leaf nodes correspond to enrollment speakers, and the red arrows show the search path of an identification task.*

We further make an empirical test. To make the test simple, we assume that each leaf node corresponds to a particular enrolled speaker and that each test OB code matches a leaf node. This means that if a leaf node is reached by a test code, and the leaf node corresponds to the correct speaker of the query code, then a Top-1 hit is triggered.

We compare three retrieval modes: (1) linear search based on cosine distance with dense vectors produced by oAE; (2) linear search based on Hamming distance with OB codes produced by obAE; (3) tree search with OB codes produced by obAE. The results on VoxCeleb1 are reported in Table 5. It can be observed that with almost the same Top-1 accuracy, tree search obtains a remarkable advantage in speed: it is about 1,300 times faster than the linear search based on cosine distance and 450 times faster than the linear search based on Hamming distance. This demonstrates that OB codes possess great potential in large-scale retrieval tasks.

Table 5: *Speed test with linear cosine search, linear Hamming search, and binary tree search. 'Speed' means the search time in millisecond and 'Top-1' represents Top-1 accuracy.*

Code	Distance	32 dims/bits		40 dims/bits		48 dims/bits	
		Speed	Top-1	Speed	Top-1	Speed	Top-1
Dense	Cosine	52.89	0.950	53.17	1.000	51.87	1.000
OB	Hamming	18.97	0.950	19.84	0.981	19.98	1.000
OB	Binary tree	0.04	0.950	0.05	0.981	0.07	1.000

5. Conclusions

In this paper, we presented an ordered binary AE (obAE) model to produce ordered binary (OB) speaker codes. The model employs nested dropout to learn ordered representations and Bernoulli sampling to get binary codes. Extensive experiments were conducted to test the performance, orderliness, convergency, and speed of the ordered binary codes, and the results showed that OB codes can bring remarkable speeding up, demonstrating the great potential of the new approach in large-scale speaker identification tasks. The future work will apply obAE to other speech processing tasks, e.g., acoustic model compression for speech recognition, and investigate task-oriented supervision.

6. References

[1] T. F. Zheng and L. Li, *Robustness-related issues in speaker recognition*. Springer, 2017, vol. 2.

[2] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[3] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[4] L. Li, Y. Chen, Y. Shi, Z. Tang, and D. Wang, "Deep speaker feature learning for text-independent speaker verification," in *INTERSPEECH*, 2017, pp. 1542–1546.

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

[6] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.

[7] S. O. Sadjadi, C. S. Greenberg, E. Singer, D. A. Reynolds, L. P. Mason, J. Hernandez-Cordero *et al.*, "The 2019 NIST speaker recognition evaluation CTS challenge." in *Odyssey*, 2020, pp. 266–272.

[8] S. O. Sadjadi, C. Greenberg, E. Singer, L. Mason, and D. Reynolds, "The 2021 NIST speaker recognition evaluation," *arXiv preprint arXiv:2204.10242*, 2022.

[9] A. Nagrani, J. S. Chung, J. Huh, A. Brown, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "VoxSRC 2020: The second voxceleb speaker recognition challenge," *arXiv preprint arXiv:2012.06867*, 2020.

[10] A. Brown, J. Huh, J. S. Chung, A. Nagrani, and A. Zisserman, "VoxSRC 2021: The third voxceleb speaker recognition challenge," *arXiv preprint arXiv:2201.04583*, 2022.

[11] D. Wang, Q. Hong, L. Li, W. Du, Y. Zhang, T. Jiang, H. Bu, and X. Xu, "CNSRC 2022 evaluation plan," 2022.

[12] V. R. Apsingekar and P. L. De Leon, "Speaker model clustering for efficient speaker identification in large population applications," *IEEE transactions on audio, speech, and language processing*, vol. 17, no. 4, pp. 848–853, 2009.

[13] Y. Hu, D. Wu, and A. Nucci, "Fuzzy-clustering-based decision tree approach for large population speaker identification," *IEEE transactions on audio, speech, and language processing*, vol. 21, no. 4, pp. 762–774, 2012.

[14] W. Jeon and Y.-M. Cheng, "Efficient speaker search over large populations using kernelized locality-sensitive hashing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4261–4264.

[15] L. Li, C. Xing, D. Wang, K. Yu, and T. F. Zheng, "Binary speaker embedding," in *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2016, pp. 1–4.

[16] O. Rippel, M. Gelbart, and R. Adams, "Learning ordered representations with nested dropout," in *International Conference on Machine Learning*. PMLR, 2014, pp. 1746–1754.

[17] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, no. 6, 1999, pp. 518–529.

[18] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.

[19] L. Li, R. Liu, J. Kang, Y. Fan, H. Cui, Y. Cai, R. Vipperla, T. F. Zheng, and D. Wang, "CN-Celeb: multi-genre speaker recognition," *Speech Communication*, vol. 137, pp. 77–91, 2022.

[20] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[21] S. Ladjal, A. Newson, and C.-H. Pham, "A pca-like autoencoder," *arXiv preprint arXiv:1904.01277*, 2019.

[22] Y. Xu, Y. Song, S. Garg, L. Gong, R. Shu, A. Grover, and S. Ermon, "Anytime sampling for autoregressive models via ordered autoencoding," *arXiv preprint arXiv:2102.11495*, 2021.

[23] Y. Shen, S. Tan, A. Sordoni, and A. Courville, "Ordered neurons: Integrating tree structures into recurrent neural networks," *arXiv preprint arXiv:1810.09536*, 2018.

[24] Y. Lu, Y. Zhu, Y. Yang, A. Said, and T. S. Cohen, "Progressive neural image compression with nested quantization and latent ordering," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 539–543.

[25] Y. Cui, Z. Liu, W. Yao, Q. Li, A. B. Chan, T.-w. Kuo, and C. J. Xue, "Fully nested neural network for adaptive compression and quantization." in *IJCAI*, 2020, pp. 2080–2087.

[26] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[27] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," in *Second international conference on learning representations, ICLR*, vol. 19, 2014, p. 121.

[28] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[29] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw)*. IEEE, 2015, pp. 1–5.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[32] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380–388.