Deep Hashing for Speaker Identification and Retrieval

Lei Fan, Qing-Yuan Jiang, Ya-Qi Yu and Wu-Jun Li

National Key Laboratory for Novel Software Technology
Collaborative Innovation Center of Novel Software Technology and Industrialization
Department of Computer Science and Technology, Nanjing University, P. R. China
{fanl,jiangqy,yuyq}@lamda.nju.edu.cn, liwujun@nju.edu.cn

Abstract

Speaker identification and retrieval have been widely used in real applications. To overcome the inefficiency problem caused by real-valued representations, there have appeared some speaker hashing methods for speaker identification and retrieval by learning binary codes as representations. However, these hashing methods are based on i-vector and cannot achieve satisfactory retrieval accuracy as they cannot learn discriminative feature representations. In this paper, we propose a novel deep hashing method, called deep additive margin hashing (DAMH), to improve retrieval performance for speaker identification and retrieval task. Compared with existing speaker hashing methods, DAMH can perform feature learning and binary code learning seamlessly by incorporating these two procedures into an end-to-end architecture. Experimental results on a large-scale audio dataset VoxCeleb2 show that DAMH can outperform existing speaker hashing methods to achieve state-of-the-art performance.

Index Terms: speaker identification and retrieval, deep hashing, additive margin softmax, deep additive margin hashing

1. Introduction

Speaker identification and retrieval [1, 2, 3] have been widely used in real applications including automatic access control of banking services, financial transactions and detection of speakers in complex scenes. Both speaker identification and retrieval can be realized by a retrieval procedure[1]. To realize the retrieval procedure, one common solution is to embed utterances into low-dimensional representations firstly, which is also called speaker embedding, and then perform retrieval based on the low-dimensional representations.

Over the past decades, real-value based speaker embedding has made good progress and achieved promising accuracy [2, 4, 5, 6]. I-vector based approaches [4], which project the Gaussian mixture model (GMM) super vector into a low-dimensional vector, have dominated the field of speaker embedding. I-vector based systems are robust and accurate in the cases with utterances of sufficient length [4]. Nevertheless, long speech isn't always available in real applications. With the upsurge of deep learning, many works have recently been devoted to deep neural networks (DNN) [2, 5, 6] and achieved promising performance due to the powerful modeling capacity of DNN. DNN based systems can outperform i-vector based systems in the case of short utterances, which is more common and practical than long utterances in real applications. Since i-vector and DNN based methods are real-value based methods, they usually suffer from high storage cost and low retrieval speed in real applications with large-scale datasets.

To enable fast query and reduce storage cost, there have appeared some hashing methods [1, 3], also called speaker hashing methods, for speaker identification and retrieval. By representing each utterance as a binary code, speaker hashing can reduce the storage cost dramatically. Furthermore, we can achieve constant or sub-linear query speed based on binary codes. However, existing speaker hashing methods [1, 3] are based on i-vector. Specifically, each utterance is represented as an i-vector in the first stage. Then the hash function is utilized to generate binary codes for utterances in the second stage. On one hand, the retrieval performance of them is limited by i-vector representations. On the other hand, existing speaker hashing methods are two-stage methods and they cannot learn optimally compatible feature for hashing. Hence, the retrieval performance of these methods is far from satisfactory in real applications.

To overcome the drawbacks of existing speaker hashing methods, in this paper we propose a novel deep hashing method, called deep additive margin hashing (DAMH). The contributions of this paper are listed as follows:

- DAMH is an end-to-end deep hashing method for speaker identification and retrieval. To the best of our knowledge, DAMH is the first deep hashing method for speaker identification and retrieval task. Compared with existing speaker hashing methods, DAMH can perform audio feature learning and binary code learning simultaneously. Hence, these two procedures can give feedback to each other.
- DAMH utilizes additive margin softmax loss to supervise speaker hashing. Angular margin added in the loss makes the learned binary codes more discriminative.
- Experiments on a large-scale audio dataset VoxCeleb2 demonstrate that DAMH can outperform existing speaker hashing methods to achieve state-of-the-art performance.

2. Related Works

In this section, we briefly review the related works, including real-value based speaker embedding and speaker hashing.

2.1. Real-value based Speaker Embedding

To perform speaker embedding, i-vector [4] was proposed to represent the GMM super vector in a single total variability space instead of two distinct spaces, i.e., speaker space and channel space. Modeling all variability as a single manifold has superior performance in this total variability model (TVM). The i-vector is the vector of latent factors which represent the speaker information of a given utterance. After the TVM model

[1] In some cases, speaker identification can also be realized by classification. In this paper, we only focus on retrieval based speaker identification.

is trained using the EM algorithm, i-vector can be extracted with the maximum a posteriori (MAP) for each utterance.

With the rise of deep learning [7], some works [2, 5, 6, 8] have been developed by using DNN for learning speaker embedding and achieved promising performance. There mainly exist two categories of DNN based speaker embedding methods. One aims to classify speakers using the frame level feature [5]. The other tries to classify speakers using the utterance level feature [2]. After training, intermediate layer features, which might be extracted from a single layer or multiple intermediate layers, are used as speaker embedding.

With the rapid growth of audio data in real applications, real-value based retrieval usually suffers from high storage cost and low query speed.

2.2. Speaker Hashing

To address the inefficiency problem in real-value based retrieval, many hashing methods have been proposed [9, 10, 11, 12]. There have appeared two speaker hashing methods [1, 3] for speaker identification and retrieval.

In [1], locality sensitive hashing (LSH) [9] over i-vectors is applied to achieve faster speaker retrieval. Specifically, a d-dimensional i-vector $\mathbf{v}$ is transformed to a lower dimensional vector with a random weight matrix $\mathbf{A} \in \mathbb{R}^{d\times K}$ drawn from a Gaussian distribution, where $K < d$. Then an element-wise sign function is adopted to turn the lower dimensional vector into the binary code $\mathbf{b}$. The binary code $\mathbf{b}$ can be calculated based on the equation: $\mathbf{b} = \text{sign}(\mathbf{A}^{\mathrm{T}}\mathbf{v})$.

Because LSH based speaker hashing method in [1] utilizes random projection matrix as hash functions, it usually requires long binary codes to achieve satisfactory accuracy. To improve the retrieval performance, Hamming distance metric learning (HDML) [13, 3] has been applied for speaker identification. HDML is a triplet-based supervised hashing method, which tries to preserve the relative similarity defined over triplet inputs like $\{\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-\}$. Here, $\{\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-\}$ is constructed based on an anchor sample $\mathbf{x}$, its similar sample $\mathbf{x}^+$ and its dissimilar sample $\mathbf{x}^-$. HDML employs triplet-based model by adopting a hinge loss over the learned triplet binary codes. In [3], HDML was applied for speaker identification and retrieval task.

Although aforementioned speaker hashing methods have been used for speaker identification and retrieval, the retrieval performance is far from satisfactory as these methods cannot fully exploit the power of feature learning. In this paper, we propose a deep hashing method, which will be presented in the following section, to perform feature learning and binary code learning seamlessly.

3. Deep Additive Margin Hashing

In this section, we present deep additive margin hashing (DAMH) in detail, including model formulation and learning algorithm.

3.1. Model

The proposed DAMH is shown in Figure 1, which consists of two components, i.e., *feature learning part* and *objective function part*.

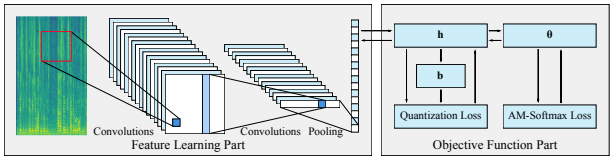

Figure 1: *The end-to-end deep learning framework of DAMH.*

Table 1: *CNN architecture modified from ResNet-34 for spectrogram input. K represents the binary code length and C represents the number of speakers in training set.*

Layer	Configuration
conv1	7×7, 64, stride 2
max pooling	3×3 max pooling, stride 2
conv2_x	$\begin{bmatrix} 3 \times 3, 64, \text{BN}, \text{ReLU} \\ 3 \times 3, 64, \text{BN} \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3, 128, \text{BN}, \text{ReLU} \\ 3 \times 3, 128, \text{BN} \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3, 256, \text{BN}, \text{ReLU} \\ 3 \times 3, 256, \text{BN} \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3 \times 3, 512, \text{BN}, \text{ReLU} \\ 3 \times 3, 512, \text{BN} \end{bmatrix} \times 3$
conv6	16×1, 512, stride 1
avg. pooling	1×1 adaptive avg. pooling, stride 1
hash layer	$512 \times K$, sign
classification layer	$K \times C$

3.1.1. Feature Learning Part

The feature learning part of DAMH model uses a convolutional neural network (CNN) architecture modified from ResNet-34 [14], which is shown in Table 1. As shown in Table 1, the CNN model contains six groups of convolutional layers, one max pooling layer, one average pooling layer, one hash layer and one classification layer. The first convolutional layer contains 64 convolution filters where the kernel size is 7×7. Following the first convolutional layer, a max pooling layer is adopted. The second to the fifth groups of convolutional layers, which contain 3, 4, 6 and 3 blocks respectively, are designed in a skip connection style. The first convolutional layer in each block is followed by a batch normalization (BN) layer and a ReLU layer successively. The second convolutional layer in each block is only followed by a BN layer. After the fifth group of convolutional layers *conv5_x*, we can get the 512 channels of $16 \times t$ intermediate feature maps, where t is determined by the length of audio segment. Then the sixth convolutional layer *conv6* is employed to combine local frequency features for each channel, followed by an adaptive average pooling layer *avg. pooling* to calculate a temporal mean of the t frames. These modifications make the model focus on frequency variance instead of temporal position. Hence, the capability of capturing speaker information is improved. After that, a hash layer transforms the feature from a 512-dimensional real-valued vector into a binary code vector of K bits. Then the binary code will be utilized as input of the classification layer.

Please note that in DAMH the architecture of ResNet-34 is used as an example for illustration, which may be replaced by

other network architectures.

3.1.2. Objective Function Part

Given an input data $\mathbf{x}_i$, we define the output of the hash layer as $\mathbf{b}_i = \text{sign}(f(\mathbf{x}_i; \Theta_{cnn})) \in \{-1, +1\}^K$, where Θ_{cnn} denotes the parameters of the CNN architecture except for the classification layer. Then we adopt the binary codes as the input of classification layer. Given N training examples, we can define the objective function with additive margin softmax (AM-Softmax) loss [15] as follows:

$$\begin{aligned} \min \quad & \mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\theta_{y_i,i}-m)}}{e^{s(\theta_{y_i,i}-m)}+\sum_{j=1,j\neq y_i}^{C}e^{s\cdot\theta_{j,i}}}, \\ s.t. \quad & \mathbf{b}_i \in \{-1,+1\}^K, \quad \forall i \in \{1,\ldots,N\}, \end{aligned} \tag{1}$$

where $y_i \in \{1, \ldots, C\}$ denotes the class label of input $\mathbf{x}_i$. $\theta_{j,i}$ is the cosine similarity between $\mathbf{W}_{*j}$ and $\mathbf{b}_i$, i.e., $\theta_{j,i} = \frac{\mathbf{W}_{*j}^{\mathrm{T}}\mathbf{b}_i}{\|\mathbf{W}_{*j}\|\|\mathbf{b}_i\|}$. Here, $\mathbf{W}$ denotes the parameters of the classification layer and $\mathbf{W}_{*j}$ is the jth column of $\mathbf{W}$, $\mathbf{b}_i$ denotes the binary code of $\mathbf{x}_i$, m is the additive margin, s is a scaling hyper-parameter. By minimizing the objective function $\mathcal{L}$ defined in problem (1), the training examples of the same class will be mapped to similar binary codes with smaller Hamming distance than that of training examples from different classes.

However, as the sign function is adopted to get the binary code in the hash layer, we cannot back-propagate the gradient to Θ_{cnn} due to the zero-gradient problem. In this paper, we utilize $\tanh(\cdot)$ to approximate $\text{sign}(\cdot)$ and rewrite problem (1) as the following form:

$$\begin{aligned} \min \quad & \widetilde{\mathcal{L}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\widetilde{\theta}_{y_i,i}-m)}}{e^{s(\widetilde{\theta}_{y_i,i}-m)}+\sum_{j=1,j\neq y_i}^{C}e^{s\cdot\widetilde{\theta}_{j,i}}} \\ & \quad + \frac{\lambda}{N}\sum_{i=1}^{N}\|\mathbf{b}_i - \mathbf{h}_i\|_2^2, \\ s.t. \quad & \mathbf{b}_i \in \{-1,+1\}^K, \quad \forall i \in \{1,\ldots,N\}, \end{aligned} \tag{2}$$

where $\mathbf{h}_i = \tanh(f(\mathbf{x}_i; \Theta_{cnn}))$, $\widetilde{\theta}_{j,i} = \frac{\mathbf{W}_{*j}^{\mathrm{T}}\mathbf{h}_i}{\|\mathbf{W}_{*j}\|\|\mathbf{h}_i\|}$, and λ is a hyper-parameter.

3.2. Learning

We adopt an alternating learning algorithm to learn binary codes $\{\mathbf{b}_i\}_{i=1}^N$ and neural network parameters $\Theta = \{\Theta_{cnn}; \mathbf{W}\}$. More specifically, we learn one group of parameters with another group of parameters fixed.

3.2.1. Update $\{\mathbf{b}_i\}_{i=1}^N$ with Θ Fixed

When Θ is fixed, we can rewrite problem (2) as follows:

$$\begin{aligned} \min \quad & \widetilde{\mathcal{L}}(\{\mathbf{b}_i\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{b}_i - \mathbf{h}_i\|_2^2 \\ & \qquad = -\frac{2}{N}\sum_{i=1}^{N}\mathbf{b}_i^{\mathrm{T}}\mathbf{h}_i + const, \\ s.t. \quad & \mathbf{b}_i \in \{-1,+1\}^K, \quad \forall i \in \{1,\ldots,N\}, \end{aligned}$$

where $const$ denotes a constant.

The elements in the binary code vector $\mathbf{b}_i$ should keep the same sign as the corresponding elements in $\mathbf{h}_i$ to maximize $\mathbf{b}_i^{\mathrm{T}}\mathbf{h}_i$. Thus we can get the following closed-form solution:

$$\mathbf{b}_i = \text{sign}(\mathbf{h}_i), \quad \forall i \in \{1,\ldots,N\}.$$

Here, $\text{sign}(\cdot)$ is an element-wise sign function.

Algorithm 1 Learning algorithm for DAMH

Input: $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$: training utterances;
$\mathbf{y} = \{y_i\}_{i=1}^N$: person identities for training utterances;
K: binary code length.
Output: Θ and $\{\mathbf{b}_i\}_{i=1}^N$.

```
Procedure
Initialize deep neural network parameters Θ, mini-batch size M and iteration number T;
for iter = 1 → T do
    for k = 1 → N/M do
        Randomly select M samples to construct a mini-batch;
        Calculate h_i by forward propagation for each x_i in the mini-batch;
        Update b_i according to b_i = sign(h_i);
        Calculate the gradient ∂L̃/∂W and ∂L̃/∂h_i according to (3) and (4);
        Calculate the gradient ∂L̃/∂Θ_cnn using chain rule;
        Update Θ based on mini-batch SGD algorithm;
        Increase margin m gradually;
    end for
end for
```

3.2.2. Update Θ with $\{\mathbf{b}_i\}_{i=1}^N$ Fixed

When $\{\mathbf{b}_i\}_{i=1}^N$ is fixed, we can utilize back-propagation to update Θ according to the following gradients:

$$\frac{\partial\widetilde{\mathcal{L}}}{\partial\mathbf{W}_{*j}} = \sum_{i=1}^{N}\left[\frac{\partial\widetilde{\mathcal{L}}}{\partial\widetilde{\theta}_{j,i}}\frac{1}{\|\mathbf{W}_{*j}\|\|\mathbf{h}_i\|}\left(\mathbf{h}_i - \mathbf{W}_{*j}^{\mathrm{T}}\mathbf{h}_i\frac{\mathbf{W}_{*j}}{\|\mathbf{W}_{*j}\|^2}\right)\right], \tag{3}$$

$$\begin{aligned} \frac{\partial\widetilde{\mathcal{L}}}{\partial\mathbf{h}_i} = & \sum_{j=1}^{C}\left[\frac{\partial\widetilde{\mathcal{L}}}{\partial\widetilde{\theta}_{j,i}}\frac{1}{\|\mathbf{W}_{*j}\|\|\mathbf{h}_i\|}\left(\mathbf{W}_{*j} - \mathbf{W}_{*j}^{\mathrm{T}}\mathbf{h}_i\frac{\mathbf{h}_i}{\|\mathbf{h}_i\|^2}\right)\right] \\ & - \frac{2\lambda}{N}(\mathbf{b}_i - \mathbf{h}_i). \end{aligned} \tag{4}$$

Then we can use chain rule to compute $\frac{\partial\mathcal{L}}{\partial\Theta_{cnn}}$. Based on the computed gradients, we utilize mini-batch stochastic gradient descent (SGD) to update Θ.

The whole learning algorithm for DAMH is summarized in Algorithm 1.

4. Experiment

To verify the effectiveness of DAMH, we carry out experiments on a workstation with an Intel (R) CPU E5-2620V4@2.1G of 8 cores, 128G RAM and an NVIDIA (R) GPU TITAN Xp.

4.1. Dataset

VoxCeleb2 [2] is a widely used dataset for speaker recognition (identification) task. We use this dataset for evaluating DAMH and baselines. VoxCeleb2 collected the utterances from YouTube videos containing thousands of speakers which span different races and a wide range of different accents. Background noise from a large number of environments and over-

Table 2: *Training/validation/test split. POI:Person of Interest.*

Dataset	# of	Training	Validation	Test
VoxCeleb2	POIs	3,641		
	utterances	903,572	18,205	36,410

lapping speech make speaker identification and retrieval challenging on this dataset.

We remove speakers whose utterance numbers are less than one hundred. The remaining 958,187 utterances from 3,641 speakers are divided into training set, validation set and test set. Validation set and test set contain five and ten utterances of each speaker respectively. Details of the training set, validation set and test set are described in Table 2.

4.2. Baselines and Evaluation Protocols

Two speaker hashing methods, LSH [9, 1] and HDML [13, 3], are selected as baselines. Besides these two methods, other hashing methods can also be utilized for speaker identification and retrieval. We choose two representative hashing methods, iterative quantization (ITQ) [10] and isotropic hashing (IsoH) [16], as baselines for comprehensive comparison.

We utilize Gaussian mixture model - universal background model (GMM-UBM) [17] to extract i-vector. Specifically, GMM-UBM uses 20-dimensional mel-frequency cepstral coefficients (MFCC) as input and extracts 400-dimensional i-vector. After that, we use linear discriminant analysis (LDA) and within-class covariance normalization (WCCN) [18] to reduce the dimensionality of i-vector to 150.

For our proposed DAMH, we randomly slice a 3-second utterance from each original utterance for training. A sliding Hamming window is used to compute the spectrogram of each utterance. Feature length, window width and step-size are set to 512, 25ms and 10ms respectively. Normalization along the axis of frequency is performed on the features. Margin m of DAMH loss function is set to a small value at the beginning and gradually increases during the training procedure. m will be fixed after reaching 0.35. s and λ are set to 30.0 and $0.1 \times \frac{1}{K}$ respectively, where K is the length of binary codes. We set the mini-batch size to 64 and tune the learning rate from 10^{-2} to 10^{-5}. Each model is trained for 36 epochs and the average training time for each epoch is 178 minutes.

We select top-1 accuracy and mean average precision (MAP) to evaluate the proposed DAMH and baselines for speaker identification and retrieval, respectively. Furthermore, we report the retrieval time and storage cost for real-value based methods and our DAMH to verify the efficiency of DAMH.

4.3. Accuracy

The top-1 accuracy for speaker identification task is presented in Table 3 with binary code length being 32, 64, 96, 128 and 256 respectively. As the dimensionality of i-vector is less than 256, the accuracy for IsoH and ITQ, which are based on principal component analysis (PCA), cannot be calculated when the binary code length is 256. Besides all hashing baselines, we also utilize two real-value based methods, i-vector and AM-Softmax, for comparison. Here, AM-Softmax denotes the method using real-valued features learned with a variant of DAMH without the binary constraint. We can see that our proposed DAMH can outperform all hashing baselines to achieve the highest accuracy. Comparing DAMH with real-value based methods, we can see that DAMH outperforms i-vector. DAMH with 256 bits can achieve comparable accuracy compared with AM-Softmax.

Table 3: *Top-1 accuracy (%) of speaker identification.*

Method	Code length				
	32	64	96	128	256
DAMH	**89.98**	**94.72**	**96.16**	**97.74**	**98.19**
IsoH	27.81	54.24	65.54	71.67	N/A
ITQ	28.88	55.82	67.19	72.93	N/A
HDML	27.58	55.48	67.25	73.03	82.33
LSH	9.13	28.01	44.31	54.23	75.80
i-vector	93.81				
AM-Softmax	98.65				

Table 4: *MAP (%) of speaker retrieval.*

Method	Code length				
	32	64	96	128	256
DAMH	**72.87**	**88.18**	**90.38**	**92.20**	**94.55**
IsoH	4.95	10.90	14.28	16.28	N/A
ITQ	5.78	12.59	16.46	18.60	N/A
HDML	6.55	14.65	43.23	49.89	61.43
LSH	0.79	2.86	5.98	8.65	18.20
i-vector	27.70				
AM-Softmax	95.82				

In Table 4, we present the MAP for speaker retrieval task. From Table 4, we can find that DAMH can outperform all hashing baselines in all cases. Furthermore, DAMH can outperform i-vector, and DAMH with 256 bits can achieve comparable MAP compared with AM-Softmax.

4.4. Efficiency

In real applications, real-value based speaker identification and retrieval methods might be impractical for massive audio data. Hashing based retrieval is used to enable fast query based on binary representation.

We report the retrieval time for DAMH and real-value based methods in Table 5. From Table 5, we can find that DAMH is faster than real-value based methods with comparable top-1 accuracy. DAMH can also reduce the storage cost compared with real-value based methods. Hence, DAMH is more practical than real-value based methods in real applications.

Table 5: *Top-1 accuracy (%), retrieval time (in second) and database storage cost (in MB) of speaker identification.*

Method	#bit/dim	Accuracy	Time	Storage cost
DAMH	64 bits	94.72	0.1327	5
	256 bits	98.19	0.1660	23
i-vector	150	93.81	0.3395	1018
AM-Softmax	512	98.65	0.6509	3539

5. Conclusion

In this paper, we propose a novel deep hashing method, called deep additive margin hashing (DAMH), for speaker identification and retrieval. To the best of our knowledge, DAMH is the first deep hashing method for speaker identification and retrieval task. Experiments on a large scale audio dataset show that our proposed DAMH can outperform baselines to achieve state-of-the-art retrieval performance.

6. Acknowledgement

This work is supported by the NSFC-NRF Joint Research Project (No. 61861146001).

7. References

[1] L. Schmidt, M. Sharifi, and I. Lopez-Moreno, "Large-scale speaker identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 1650–1654.

[2] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Annual Conference of the International Speech Communication Association*, 2018, pp. 1086–1090.

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

[4] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[5] E. Variani, X. Lei, E. McDermott, I. Lopez-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*, 2014, pp. 4052–4056.

[6] 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*, 2018, pp. 5329–5333.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[8] Y.-Q. Yu, L. Fan, and W.-J. Li, "Ensemble additive margin softmax for speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 6046–6050.

[9] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *ACM Symposium on Computational Geometry*, 2004, pp. 253–262.

[10] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 817–824.

[11] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 1711–1717.

[12] Q.-Y. Jiang, X. Cui, and W.-J. Li, "Deep discrete supervised hashing," *IEEE Transaction Image Processing*, vol. 27, no. 12, pp. 5996–6009, 2018.

[13] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 1070–1078.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[15] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letter*, vol. 25, no. 7, pp. 926–930, 2018.

[16] W. Kong and W.-J. Li, "Isotropic hashing," in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 1655–1663.

[17] D. Gutman and Y. Bistritz, "Speaker verification using phoneme-adapted gaussian mixture models," in *European Signal Processing Conference*, 2002, pp. 1–4.

[18] A. O. Hatch, S. S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for svm-based speaker recognition," in *International Conference on Spoken Language Processing*, 2006, pp. 1471–1474.