SIMULTANEOUSLY LEARNING ROBUST AUDIO EMBEDDINGS AND BALANCED HASH CODES FOR QUERY-BY-EXAMPLE

Anup Singh[†⋆] *Kris Demuynck*[†] *Vipul Arora*[⋆]

[⋆] Department of Electrical Engineering, Indian Institute of Technology Kanpur, India
[†] IDLab, Department of Electronics and Information Systems, imec - Ghent University, Belgium

ABSTRACT

Audio fingerprinting systems must efficiently and robustly identify query snippets in an extensive database. To this end, state-of-the-art systems use deep learning to generate compact audio fingerprints. These systems deploy indexing methods, which quantize fingerprints to hash codes in an unsupervised manner to expedite the search. However, these methods generate imbalanced hash codes, leading to their suboptimal performance. Therefore, we propose a self-supervised learning framework to compute fingerprints and balanced hash codes in an end-to-end manner to achieve both fast and accurate retrieval performance. We model hash codes as a balanced clustering process, which we regard as an instance of the optimal transport problem. Experimental results indicate that the proposed approach improves retrieval efficiency while preserving high accuracy, particularly at high distortion levels, compared to the competing methods. Moreover, our system is efficient and scalable in computational load and memory storage.

Index Terms— Audio fingerprinting, optimal transport, hashing, efficient retrieval, self-supervised learning

1. INTRODUCTION

Audio fingerprinting transforms audio signal into a compact and unique representation, thereby enabling efficient identification of query audio in a reference database. With digital devices becoming ubiquitous, there are several possibilities for audio fingerprinting applications, ranging from music identification based on query-by-example [1, 2], second-screen services [3], broadcast monitoring [4] to automated indexing of large-scale multimedia archives.

In general, the audio fingerprinting system must be robust against various distortions, generate discriminative fingerprints for perceptually different audio segments, and be computationally efficient. The conventional approaches [5–7] generate binary fingerprints based on spectral-temporal features to expedite the search. However, these approaches are computationally expensive and perform poorly at high distortion levels. One popular and exemplary system of a traditional approach is Shazam [5].

Deep learning has recently been investigated for robust audio embeddings. Google's music recognizer [8] generates compact audio embeddings by training a CNN model using the triplet loss function. NAFP [9] trains a similar model as [8] using the contrastive learning framework and focuses on precisely locating a query in the identified audio. However, these approaches use unsupervised indexing methods, which hamper their retrieval performance. For instance, Locality-Sensitive Hashing (LSH) [10] reduces the search space by transforming high-dimensional embeddings into compact binary codes and indexing them using multiple hash tables. But, this method is ineffective for large-scale deployment since unsupervised mapping is prone to being non-discriminative and/or noise-sensitive, thus requiring multiple costly probes of hash buckets to achieve satisfactory performance.

Therefore, simultaneously learning audio embeddings and binary encodings is a promising solution for improving retrieval effectiveness. However, learning binary encoding is a discrete learning process, making end-to-end training challenging. In addition, a uniform distribution (code balance) of binary encodings is required to facilitate an efficient search in the Hamming space. Existing works on learned hash functions [11–13] tend to suffer from the code imbalance problem, resulting in sub-optimal (slow) retrieval performance.

This paper presents an audio fingerprinting system that is robust against high noise and reverberation levels and performs a comprehensive audio search. Overall, the main contributions of our work are as follows:

- An end-to-end self-supervised learning framework for jointly learning continuous and discrete audio embeddings (binary encodings). To our knowledge, no prior work exists for the audio retrieval task.
- Adopting the optimal transport problem to our work to achieve balanced hash codes and improve retrieval efficiency.

2. METHOD

This section describes our framework, which comprises sequentially stacked modules to generate robust continuous and discrete embeddings end-to-end via deploying self-supervised learning and a balanced clustering constraint. An overview of our framework can be seen in Figure 1.

2.1. Feature Encoder

Given a set $\mathcal{D}$ of audio files, we randomly select an audio segment of t-seconds, denoted as x. We construct a distorted counterpart x^+ of x by applying stochastic distortions to it, resulting in an input pair $\{x, x^+\}$ for our contrastive learning framework. Next, these segments are individually transformed into a log Mel spectrogram using a hamming window of length 25 ms and hop length of 10 ms. The spectrogram is then subdivided into T non-overlapping patches and fed to a feature encoder, which consists of 3 sequentially stacked modules: the patch embedding layer, a transformer-based encoder, and a projection layer, to generate d-dimensional audio embeddings. The goal of the encoder is to learn a mapping $\mathcal{F}_\theta$ such that $\mathcal{F}_\theta(x)$ and $\mathcal{F}_\theta(x^+)$ are closer to each other in the $L^2(\mathbb{R}^d)$ space using the contrastive loss function:

$$\mathcal{L}_c = -\log \frac{e^{(\mathcal{F}_\theta(x)\cdot\mathcal{F}_\theta(x^+))/\tau}}{e^{(\mathcal{F}_\theta(x)\cdot\mathcal{F}_\theta(x^+))/\tau} + \sum_{x^-} e^{(\mathcal{F}_\theta(x)\cdot\mathcal{F}_\theta(x^-))/\tau}} \quad (1)$$

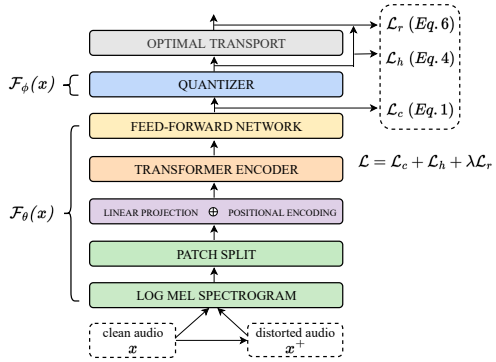

Fig. 1. An illustration of the proposed framework to jointly learn continuous embeddings and hash codes for a t-sec input audio segment. We use contrastive loss on encoder and quantizer outputs in conjunction with clustering loss to learn robust audio representations and balanced encodings.

where τ is a temperature hyperparameter to facilitate learning from hard negatives. Here, for each pair of $\{x, x^+\}$ in the mini-batch of size N(N/2 pairs), we consider the remaining samples in the batch as negative samples, x^-, which are N-2.

2.2. Quantizer

The output of the feature encoder is fed into the quantizer to compute its K-dimensional embedding $q(x) = \mathcal{F}_\phi(\mathcal{F}_\theta(x))$. Note that training a quantizer with discrete outputs becomes problematic due to non-differentiable operations. Hence, we relax the quantizer output to include continuous values in the $L^2(\mathbb{R}^K)$ space and treat the quantization as a clustering process:

$$\mathcal{L}_r = -\max_k \big[\text{softmax}(\frac{s_k(x)}{\tau'}) + \text{softmax}(\frac{s_k(x^+)}{\tau'})\big],$$
$$\text{where } s_k(x) = q(x) \cdot \frac{\tilde{h}_k}{||\tilde{h}_k||_2}, \quad k = 1, 2, .., 2^K \tag{2}$$

Here, τ' is a temperature hyperparameter and $s_k(x)$ denotes cosine similarity of $q(x)$ with all possible 2^K K-bits hash codes $\tilde{h}_k \in [-1, 1]^K$. Minimizing $\mathcal{L}_r$ requires $q(x)$ and $q(x^+)$ to side with the direction of one of the fixed cluster centroids $\tilde{h}_k$. During inference, $q(x)$ is encoded into a K-bits hash code as:

$$h(x) = \max_{\tilde{h}_k} s_k(x) \tag{3}$$

Moreover, to generate robust hash codes, we deploy the contrastive loss on the quantizer outputs as well:

$$\mathcal{L}_h = -\log \frac{e^{(q(x)\cdot q(x^+))/\tau}}{e^{(q(x)\cdot q(x^+))/\tau} + \sum_{x^-} e^{(q(x)\cdot q(x^-))/\tau}} \tag{4}$$

2.3. Balanced clustering

It is important to emphasize that clustering high-dimensional samples map most of the samples to a few clusters, leaving other clusters underutilized. This problem adversely affects the hash-based retrieval algorithms. Moreover, a neural network may lead to a trivial solution for Eq. 2 where all quantizer outputs collapse to a single cluster centroid. Therefore, we add a balanced clustering constraint to our formulation to yield balanced hash codes:

$$\max_A \mathbb{E}_x\big[\sum_{k=1}^{2^K} A_k(x)s_k(x)\big] \quad \text{subject to}$$
$$A_k(x) \in \{0, 1\},\ \sum_{k=1}^{2^K} A_k(x) = 1\ \forall x\ ,\ \mathbb{E}_x[A_k(x)] = \frac{1}{2^K}\ \forall k \tag{5}$$

where $A_k(x)$ indicates that $q(x)$ is assigned to hash code $\tilde{h}_k$. Note that Eq. 5 is particularly difficult to solve since it is a combinatorial optimization problem with millions of samples. Hence, we derive an approximate solution by focusing on uniform clustering of a subset of $\mathcal{D}$. In addition, we consider the above formulation as the instance of an optimal transport problem, which can be efficiently solved in a nearly linear time using the Sinkhorn-Knopp algorithm [14]. We calculate the cost of mapping i^{th} sample $q(x_i)$ to $\tilde{h}_k$ as $s_k(x_i)$. Henceforth, we reformulate Eq. 2 as:

$$\mathcal{L}_r = -\big[\text{softmax}\big(\frac{s_{k^*}(x)}{\tau'}\big) + \text{softmax}\big(\frac{s_{k^*}(x^+)}{\tau'}\big)\big],$$
$$\text{where } k^* = \arg\max_k A_k(x) \tag{6}$$

Finally, we train our model using the above-formulated loss functions:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_h + \lambda\mathcal{L}_r \tag{7}$$

where the parameter $\lambda \geq 0$ controls the trade-off between clustering loss and contrastive loss.

2.4. Audio retrieval

Let the fingerprint database be represented as $\{\{f_{a,n}\}_{n=1}^N\}_{a=1}^A$, where A is the total number of reference audio tracks in the database and N is the total fingerprints generated for each audio track.

We first generate M overlapping segments for a given audio query and feed them to our model to compute their corresponding continuous embeddings, $\{q_m\}_{m=1}^M$, and hash codes. Further, these hash codes serve as keys to hash buckets in a hash table, which returns C, a set of search candidates represented by their indices (a, n). Now, we find the best-matched candidate for each q_m as:

$$a(m), n(m) = \arg\max_{(a,n)\ \in\ C} (q_m \cdot f_{a,n}) \tag{8}$$

To perform a coarse search, i.e., audio identification, we determine a with maximum evidence as the best-matched audio track.

$$a^* = \text{mode}[a(m)]_{m=1}^M, \tag{9}$$

Next, we perform a fine-grained search to precisely locate the query in an identified audio track a^* as follows:

$$m^* = \arg\min_m \sum_{l=-m}^{M-m-1} |n(m+l) - (n(m)+l)|$$
$$n^* = n(m^*) - m^* \tag{10}$$

Finally, we determine the time stamp corresponding to the index n^* as the query time stamp in the identified audio.

3. EXPERIMENTAL SETUP

3.1. Databases

- **Free Music Archive** [15]: The subset *fma_large* was used for model training, while *fma_medium* was utilized to build the reference database. There are 106K and 25K 30s audio snippets in the *fma_large* and *fma_medium*, respectively. In addition, the distorted queries were generated using the *fma_medium* subset.
- **Noises**: For model training, we extracted noises from the *MUSAN* [16] corpus, while for system testing, we used a different set of noises (*Babble, Cafeteria, Car, Living room, Shopping, Train station and Traffic*) from the *ETSI*[1] database.
- **RIRs**: Different RIRs corresponding to different room layouts were extracted from the *MUSAN* corpus for training. We chose six RIRS from the Aachen Impulse Response Database [17] with t60 values ranging from 0.1s to 0.8s for system testing.

3.2. Data Augmentation Pipeline

We construct a positive sample x^+ for each anchor sample x by stochastically applying the following audio distortions:

- **Noise mixing**: A random background noise is added in the range of 0-25dB SNR.
- **Reverberation**: The input audio is filtered with a randomly selected RIR to simulate the room acoustics.
- **Time offset**: A temporal shift of up to 50ms is added to deal with potential temporal irregularities in the real-world scenario.

3.3. Implementation details

We compared our method with the Neural audio fingerprint (NAFP) [9] system. To the best of our knowledge, NAFP is the only method that uses deep learning to generate robust audio fingerprints and outperforms earlier approaches. We also compared it to the Shazam approach used by Audfprint[2].

We trained our model with the Adam [18] optimizer for 300 epochs using a learning rate of 1e-4. The model was trained on a single NVIDIA A100 GPU for about 4 days. The NAFP model was trained using the log Mel spectrograms for 150 epochs with the same training hyperparameters. Table 1 lists the configuration of the experiments used for developing our audio fingerprinting system as well as the baseline methods.

The cost matrix for the optimal transport framework is memory intensive and grows exponentially with K, becoming a bottleneck during training. As a result, we choose 2^c cluster centroids at random for each training step and solve the balanced clustering problem. In this case, we chose c equals 12.

We built a reference database of $\sim$7M fingerprints with the *fma_medium* dataset. We utilized the LSH implementation[3] and a Hash Table (HT) to index the fingerprints generated using NAFP and our Feature Encoder (FE), respectively. To make fair comparisons between LSH and our retrieval approach, we kept the parameters of both indexers such that each fingerprint is encoded with 16 bits and allowed a maximum of 1000 probes of hash buckets.

[1] https://docbox.etsi.org/

[2] https://github.com/dpwe/audfprint

[3] https://github.com/FALCONN-LIB/FALCONN

Parameter	Value
Sampling rate (F_s)	16 kHz
Audio segment length (t)	1s
Spectrogram patches (T)	10
log Mel spectrogram dimensions	64×96
Hop length (h)	100 ms
continuous embedding dimensions (d)	128
Discrete embedding dimensions (K)	16
Batch size (N)	8192
Temperatures (τ, τ')	0.1, 0.1
λ	0.5
LSH configuration:	
Tables	10
Hash bits	16
Number of probes	1000

Table 1. Experiments configurations

3.4. Evaluation metric

We used the following metric for system evaluation at audio (coarse search) and segment (fine search) level retrieval:

$$accuracy = \frac{n\ hits\ @\ top\text{-}1}{n\ hits\ @\ top\text{-}1 + n\ miss\ @\ top\text{-}1} \times 100 \qquad (11)$$

Note that a match is declared correct for the segment level search if the located timestamp of the query in the correct retrieved audio is within $\pm$ 50 ms.

3.5. Database creation and indexing

We employ our trained model as a fingerprinter to extract fingerprints from the reference database. The fingerprints for a given audio track are generated as follows: First, we create overlapping segments of length t-seconds at h-seconds intervals. These segments are transformed into log Mel spectrograms and fed into the model, which generates their continuous audio embeddings and their discrete counterpart, viz., hash codes. These outputs comprise the fingerprints database, which is then indexed using the hash table.

3.6. Search query

We used the *fma_medium* dataset to generate 1000 search queries randomly extracted from audio tracks. We test our system with the queries distorted by noise, reverberation and a combination of both. We create noisy reverberant queries by first convolving both audio and noise signals with the RIR corresponding to a t60 level of 0.5s, followed by adding both signals at SNR levels ranging from 0dB to 25dB. In addition, we created queries of lengths: 1s, 2s, 3s and 5s to test our system performance for varying lengths.

4. RESULTS AND DISCUSSION

In Table 2, we show the retrieval accuracies and computational load in different settings.

Efficacy: Our system achieves substantially better retrieval accuracies than the NAFP system in every distortion environment for different query lengths. For instance, our system is about 20% more accurate at 0dB SNR in a noisy reverberant environment. Two factors contribute to the performance margin. First, the transformer encoder provides contextual embeddings, generating discriminative

		Noise					Noise + Reverb					Reverb					
Length		0	5	10	15	20	0	5	10	15	20	0.2	0.4	0.5	0.7	0.8	speedup
1	NAFP + LSH	50.7	69.7	73.7	76.0	76.9	20.8	43.9	55.5	58.5	58.9	62.5	61.5	58.9	49.3	45.0	1x
	FE + LSH	**66.6**	**82.6**	87.6	90.0	90.6	**44.8**	**62.6**	73.8	**79.4**	**82.0**	83.8	83.8	78.4	**69.4**	64.4	1x
	FE + HT (Ours)	64.9	81.7	**87.9**	**90.1**	**90.9**	41.9	62.0	**74.6**	79.2	81.2	**84.6**	**82.3**	**80.0**	68.7	**64.5**	**2.4x**
2	NAFP + LSH	71.0	83.4	85.5	87.6	87.5	37.8	65.6	73.5	75.2	76.7	78.6	76.4	75.7	68.0	59.5	1x
	FE + LSH	**80.4**	88.2	91.6	93.2	94.8	**63.4**	78.2	84.6	86.0	85.4	90.4	86.4	85.0	74.8	67.8	1x
	FE+HT(Ours)	80.2	**89.4**	**93.1**	**94.1**	**94.9**	61.4	**80.8**	**85.0**	**87.5**	**88.3**	**92.4**	**87.7**	**86.4**	**77.3**	**70.7**	**2.4x**
3	NAFP + LSH	77.7	84.8	88.9	89.2	89.1	50.1	72.6	80.0	79.5	80.1	83.6	79.9	77.9	70.2	63.8	1x
	FE + LSH	83.2	88.4	92.6	94.4	95.2	**71.6**	82.6	85.6	86.2	87.4	91.8	87.8	85.6	74.4	68.4	1x
	FE+HT(Ours)	**84.7**	**90.8**	**95.5**	**96.3**	**97.1**	70.7	**84.1**	**88.6**	**89.0**	**90.1**	**93.8**	**90.3**	**87.9**	**77.9**	**71.6**	**2.3x**
5	NAFP + LSH	82.6	89.2	90.2	90.5	91.2	60.2	79.1	83.4	82.8	83.1	85.6	83.4	79.3	74.1	65.7	1x
	FE + LSH	85.6	90.0	92.8	94.2	95.8	80.0	87.0	87.1	87.6	87.2	93.8	88.8	86.6	75.0	69.0	1x
	FE+HT(Ours)	**88.0**	**93.4**	**95.3**	**96.2**	**97.4**	**80.6**	**88.6**	**90.8**	**91.3**	**91.5**	**96.4**	**91.1**	**88.9**	**78.7**	**71.2**	**2.4x**

Table 2. Top1-hit rate performance(%) comparison in the segment-level search in different distortion environments for varied query lengths. The values in the last column shows the speedup achieved by the proposed method compared to LSH.

embeddings corresponding to adjacent audio segments with a similar timbre but different temporal evolution. Secondly, our balanced cluster approach enables lesser comparisons with reference samples to find the best match compared to LSH.

In all distortion settings, the retrieval accuracy improves as query length increases. The retrieval accuracies of both systems improve by 15-20% when query length is increased by 1s, particularly at high distortion levels. However, the performance gain with increasing query length is less than 2% at low distortion levels. As a result, increasing query length is particularly useful in high-distortion conditions. The obtained results demonstrate the effectiveness of our proposed audio retrieval process.

With added noise and reverberation, all systems fail to deliver reasonable performance with 1s queries, particularly at 0dB and 5dB SNR. Nevertheless, our system provides good results given longer query lengths.

Efficiency: The proposed retrieval algorithm has two steps: hash-based matching (locating hash bucket) and brute-force matching. The computational complexity of the first step is $\mathcal{O}(1)$ and that of the second step is $\mathcal{O}(N)$ with respect to the number N of candidates retrieved in the first step. To compare the efficiency of the proposed balanced hashing with LSH, we compare the number of candidates retrieved by each hashing scheme in the first step. Since the computational load scales linearly with N, we report the relative speed up for the hashing algorithm HA as $N_{\text{HA}}/N_{\text{LSH}}$.

The results show that our method achieves about a 2.4-fold speedup compared to LSH, indicating that our system is efficient while maintaining retrieval accuracy. Using LSH as an indexer improves the retrieval accuracy of our system by 1-3%, particularly in noisy reverberant conditions at 0dB SNR. However, the increase in accuracy comes at the expense of two-fold sample comparisons. Furthermore, the accuracy gap shrinks to less than 1% with the increase in query length, and our system even delivers better performance than LSH with lesser comparisons at higher SNRs ($\geq$5dB).

Due to the space limit, a plot illustrating the uniform distribution of hash codes across hash buckets can be found at webpage[4]. This shows that our hash table is balanced.

Memory requirement and search time: The memory requirement for our system equals the memory to store the fingerprints database and the hash table. For our experiments, we store fingerprints as 128-dimensional 32-bit float values and create a hash table using 16 bits hash codes. The final size of the database containing $\sim$7M fingerprints is approximately 4.8GB, while the hash table consumes 255MB.

[4]https://github.com/madhavlab/2022_icassp_sanup

Distortion	Method	0dB	5dB	10dB	15dB
Noise	Audfprint	72.1	82.7	89.4	91.2
	Ours	**93.3**	**97.1**	**98.7**	**99.2**
Noise+ Reverb	Audfprint	64.8	79.4	87.2	92.3
	Ours	**86.7**	**96.5**	**98.5**	**98.7**
		0.2s	0.4s	0.5s	0.7s	0.8s
Reverb	Audfprint	**96.1**	**94.6**	81.8	89.6	40.2
	Ours	95.0	94.4	**94.8**	**92.1**	**90.5**

Table 3. Top-1 hit rate (%) performance in the audio-level search in different distortion conditions.

We also compared our system to the Audfprint. Audfprint performs poorly in the segment-level search, especially at high distortion levels. As a result, we solely compare its audio identification performance to ours. Furthermore, its performance degrades with small query lengths; thus, the retrieval results with 5s audio query snippets are presented in Table 3. Our system demonstrates excellent retrieval accuracy, with over 95% accuracy under strong distortion settings. On the contrary, the performance of the Audfprint method drops significantly at high distortion levels, indicating that our method outperforms the conventional approach for the audio identification problem.

5. CONCLUSION

This paper presents a robust audio fingerprinting system that improves search efficiency by jointly learning continuous and discrete audio embeddings (hash codes). Discrete embeddings are modeled as a clustering process using K-bits hash codes as fixed cluster centroids. To prevent under/over-utilization of clusters, a uniform clustering constraint is introduced using the optimal transport problem. The system uses a simple and efficient sequence search strategy to estimate the query time stamp. Extensive experiments demonstrate the superior performance of our method compared to NAFP and Audfprint. The proposed approach makes the system computationally and memory-efficient, enabling deployment to big databases.

6. ACKNOWLEGMENT

This work was supported by the research grant (PB/EE/2021128-B) from Prasar Bharati. We also thank C-DAC Param Siddhi for their GPU computing resources.

7. REFERENCES

[1] "Shazam music recognition service," http://www.shazam.com/, (Last accessed: 20.01.2023).

[2] "Soundhound," http://www.soundhound.com/, (Last accessed: 20.01.2023).

[3] Christopher Howson, Eric Gautier, Philippe Gilberton, Anthony Laurent, and Yvon Legallais, "Second screen tv synchronization," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2011, pp. 361–365.

[4] Seungjae Lee and Jin S. Seo, "A tv commercial monitoring system using audio fingerprinting," in *Proceedings of the 6th International Conference on Entertainment Computing*, 2007, ICEC'07, pp. 454–457.

[5] Avery Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[6] Jaap Haitsma and Ton Kalker, "A highly robust audio fingerprinting system.," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2002, vol. 2002, pp. 107–115.

[7] Shumeet Baluja and Michele Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern recognition*, vol. 41, no. 11, pp. 3467–3480, 2008.

[8] B. Gfeller et al., "Now playing: Continuous low-power music recognition," in *Advances in Neural Information Processing Systems 2017 Workshop: Machine Learning on the Phone and other Consumer Devices*, 2017.

[9] Sungkyun Chang, Donmoon Lee, Jeongsoo Park, Hyungui Lim, Kyogu Lee, Karam Ko, and Yoonchang Han, "Neural audio fingerprint for high-specific audio retrieval based on contrastive learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3025–3029.

[10] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253–262.

[11] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang, "Deep cauchy hashing for hamming space retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1229–1237.

[12] Tiezheng Ge, Kaiming He, and Jian Sun, "Graph cuts for supervised binary coding," in *Computer Vision–ECCV 2014: 13th European Conference, Proceedings, Part VII 13*. Springer, 2014, pp. 250–264.

[13] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2014, vol. 28.

[14] Marco Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013, vol. 2, pp. 2292–2300.

[15] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson, "FMA: A dataset for music analysis," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[16] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[17] Marco Jeub, Magnus Schafer, and Peter Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *16th International Conference on Digital Signal Processing*. IEEE, 2009, pp. 1–5.

[18] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.