FlowHash: Accelerating Audio Search with Balanced Hashing via Normalizing Flow

Anup Singh, Kris Demuynck, Vipul Arora

***Abstract*—Nearest neighbor search on context representation vectors is a formidable task due to challenges posed by high dimensionality, scalability issues, and potential noise within query vectors. Our novel approach leverages normalizing flow within a self-supervised learning framework to effectively tackle these challenges, specifically in the context of audio fingerprinting tasks. Audio fingerprinting systems incorporate two key components: audio encoding and indexing. The existing systems consider these components independently, resulting in suboptimal performance. Our approach optimizes the interplay between these components, facilitating the adaptation of vectors to the indexing structure. Additionally, we distribute vectors in the latent $\mathbb{R}^K$ space using normalizing flow, resulting in balanced K-bit hash codes. This allows indexing vectors using a balanced hash table, where vectors are uniformly distributed across all possible 2^K hash buckets. This significantly accelerates retrieval, achieving speedups of up to 2$\times$ and 1.4$\times$ compared to the Locality-Sensitive Hashing (LSH) and Product Quantization (PQ), respectively. We empirically demonstrate that our system is scalable, highly effective, and efficient in identifying short audio queries ($\leq$2s), particularly at high noise and reverberation levels.**

***Index Terms*—Audio fingerprinting, Indexing, Normalizing Flows, Information Retrieval, Self-supervised learning**

I. Introduction

In the ever-expanding landscape of multimedia content, vector search has become increasingly crucial for efficiently retrieving similar items based on content representations, often represented as vectors in high-dimensional spaces. While extensive research has been devoted to content-based retrieval in the image domain [1], the domain of audio retrieval, particularly for the audio fingerprinting tasks, still needs to be explored. Audio fingerprinting generates a content-based compact summary of an audio signal, facilitating efficient storage and retrieval of the audio content. One prominent application is in the field of music recognition [2], where audio fingerprinting plays a pivotal role in identifying songs and providing information about the artist, album, and track. In the realm of copyright enforcement, audio fingerprinting becomes an indispensable tool [3] in monitoring and detecting unauthorized use or distribution of copyrighted audio material. This application is vital for content creators and rights holders who seek to protect their intellectual property in the digital landscape. Furthermore, audio fingerprinting plays a crucial role in second-screen services [4]. In scenarios where viewers engage with content on multiple screens simultaneously, such as interactive television or streaming services, audio fingerprinting facilitates synchronized and context-aware experiences. This can enhance user engagement and provide innovative ways for audiences to interact with audio content in real-time.

The existing audio fingerprinting methods rely on efficient indexing algorithms [5], [6], [7]. In particular, Locality-Sensitive Hashing (LSH) [7] implicitly divides the space into lattices, such that similar points are grouped into the same hash bucket. During retrieval, LSH evaluates the similarity between the query and points in the same hash bucket. However, when applied to real-world data characterized by non-uniform distributions, LSH faces challenges. This includes the issue of unbalanced hashing [8], where some buckets remain empty while others become overfilled, leading to reduced retrieval accuracy and unexpected delays. Furthermore, LSH, being an unsupervised method, requires multiple bucket probes and hash table constructions to achieve satisfactory performance.

In contrast to conventional approaches that consider representation learning and indexing separate processes, we introduced an approach [9] combining both aspects, which simultaneously learns robust representations and balanced hash codes. This approach achieves balanced hash buckets by solving a balanced clustering objective, utilizing the optimal transport (OT) formulation [10]. However, when dealing with a large number of cluster centroids, this requires a very large transportation matrix. Consequently, the Sinkhorn-Knopp (SK) algorithm [11], used to solve regularized transport problems, exhibits slow convergence and introduces substantial overhead during training. Also, this approach defines cluster centroids as normalized binary vectors on a hypersphere. However, these non-orthogonal centroids lead to overlapping or closely situated clusters, resulting in hash buckets lacking distinct separation. Consequently, it causes data points near bucket boundaries and their perturbations to be assigned to different hash buckets, thereby compromising retrieval performance.

While numerous prior studies [12], [13], [14] have explored the generation of balanced hash codes in the context of image retrieval, none of these algorithms addresses the utilization of all possible hash buckets for a given bit length while maintaining a balanced distribution across hash buckets.

Similar to [9], we adopt a joint representation learning and indexing approach in this paper. In contrast to [9], we use a novel method for learning balanced hash codes, namely normalizing flow (NF). NF is a generative model that

This work was supported by Prasar Bharti under Grant PB/EE/2021128B

K. Demuynck is with IDLab-imec, Department of Electronics and Information Systems, Ghent University, 9000 Ghent, Belgium (e-mail: kris.demuynck@ugent.be)

A.Singh and V. Arora are with Department of Electrical Engineering, Indian Institute of Technology, Kanpur 208016, India (e-mail: sanup@iitk.ac.in; vipular@iitk.ac.in). A.Singh is also affiliated with Ghent University, Belgium

transforms complex distributions into tractable ones through a series of invertible and differentiable mappings. Our approach employs the RealNVP [15] normalizing flow model to transform pre-trained audio representations within an $\mathbb{R}^K$ space to assign each dimension to a bimodal Gaussian mixture distribution. The result is an overall distribution comprising 2^K balanced modes, where each mode effectively corresponds to a representative hash bucket. This methodology aligns with the balanced clustering objective, with cluster centroids being vertices of a K-dimensional hypercube to ensure distinct cluster separation. Furthermore, we introduce a cross-entropy-based loss as a regularization term, enhancing the likelihood of data points and their perturbations being assigned to the same hash bucket. Our model, named **FlowHash**, presents two noteworthy advantages: firstly, it provides a scalable and optimal solution to achieve balanced hash buckets as often required in hash-based indexing. Secondly, it facilitates efficient database indexing using a classical hash table, which diverges from the prevalent method of constructing multiple hash tables in LSH. As a result, our approach enhances both the effectiveness and efficiency of the retrieval process. Overall, the main contributions of our work are as follows:

- We introduce a novel application of NF in hash-based indexing, leveraging it to obtain scalable balanced hash codes that ensure uniform distribution across all possible hash buckets for a given bit length.
- We introduce a regularization loss term to enhance the robustness of the hash codes.
- We present an audio fingerprinting method to effectively and efficiently identify short audio snippets ($\leq$2s) in high noise and reverberant environments.
- Our emperical study shows the robustness and scalability of our approach. Furthermore, we demonstrate its superior performance compared to LSH, PQ, and the recently introduced OT-based method [9].

II. Related Work

A. Audio fingerprinting

The common approaches for audio fingerprinting transform audio segments into low-dimensional vectors, commonly referred to as *representations* or *fingerprints*. The existing methods can be categorized based on two key characteristics. Firstly, the approach employed to generate fingerprints can be either knowledge-based [16], [2], [17] or machine-learning-based [18], [19]. Secondly, the generated fingerprints can be either hash representations [20], [21] or real-value representations [19], [22]. The real-value representations offer a notable advantage in precise identification owing to their expansive information-capturing capability. However, hash-based representations facilitate fast comparisons and impose lower memory demands.

The fingerprinting method introduced by Wang *et al.* [2], commonly referred to as Shazam, is a widely known approach. It captures distinctive peaks within the audio spectrogram under the assumption of their resilience to audio distortions. These discerned peaks transform hashes to facilitate fast search. Philips system [16] is another widely known approach, which generates binary fingerprints based on energy changes across spectral-temporal space. Waveprint [20] computes binary fingerprints utilizing top-k wavelets and transforms them into compact representations using the Min-Hash technique. The system proposed in [23] employs pseudo-sinusoidal components to derive fingerprints that exhibit resilience against noise and reverberation. These systems rely on handcrafted features that make it challenging to identify queries accurately in high noise and reverberation settings [22]. Moreover, these systems require long query lengths ($\geq$ 5s) to deliver accurate results in high distortion environments.

Recent developments in audio fingerprinting have been marked by the emergence of methodologies based on deep learning, particularly within the realm of self-supervised learning [24], [22], [9]. [24] trains an encoder, drawing inspiration from Google's music recognizer system [18], within the context of a contrastive learning framework. Expanding upon this initial work, [22] introduces spectral-temporal attention mechanisms into intermediate CNN features, enabling the model to learn salient patches and generate robust audio fingerprints. In a recent study, [9] introduced a system leveraging a transformer encoder to acquire contextual representations, resulting in more discriminative fingerprints. These methods exhibit robustness and achieve high retrieval performance even under high-distortion conditions such as noise and reverberation compared to conventional systems.

B. Hash-based indexing

A large body of work exists on hash-based indexes for approximate nearest-neighbor searches in high-dimensional space. Locality-sensitive hashing methods, like Multi-Probe LSH [25] and Cross-polytope LSH [26], utilize hash functions to map data to fixed-size hash codes, yielding sub-linear time complexity. The LSB-tree [27] merges LSH with trees for logarithmic query complexity. Entropy-LSH [28] improves similarity-preserving hashing through entropy-based techniques. Bayesian LSH [29] enhances search precision via probabilistic models. Deep learning-based LSH approaches, such as Deep-LSH [8] and Neural-LSH [30], leverage deep learning for effective hash function learning.

C. Generative modeling

Generative modeling plays a pivotal role in data generation and latent space exploration. The two prominent generative models include Generative Adversarial Networks (GANs) [31] and Variational Autoencoders (VAEs) [32], facilitating data generation and latent space exploration. However, neither of them evaluates explicit probability estimation on generated samples. In response to this limitation, Normalizing Flows (NFs) have emerged as a promising alternative. They employ invertible transformations to model complex distributions, yielding both tractable likelihood estimation and efficient sampling mechanism. The versatility of Normalizing Flows is evident in their diverse applications. One notable domain is density estimation [15], where NFs demonstrate efficacy in accurately modeling and capturing the underlying probability distribution of the data. Additionally, NFs contribute to audio

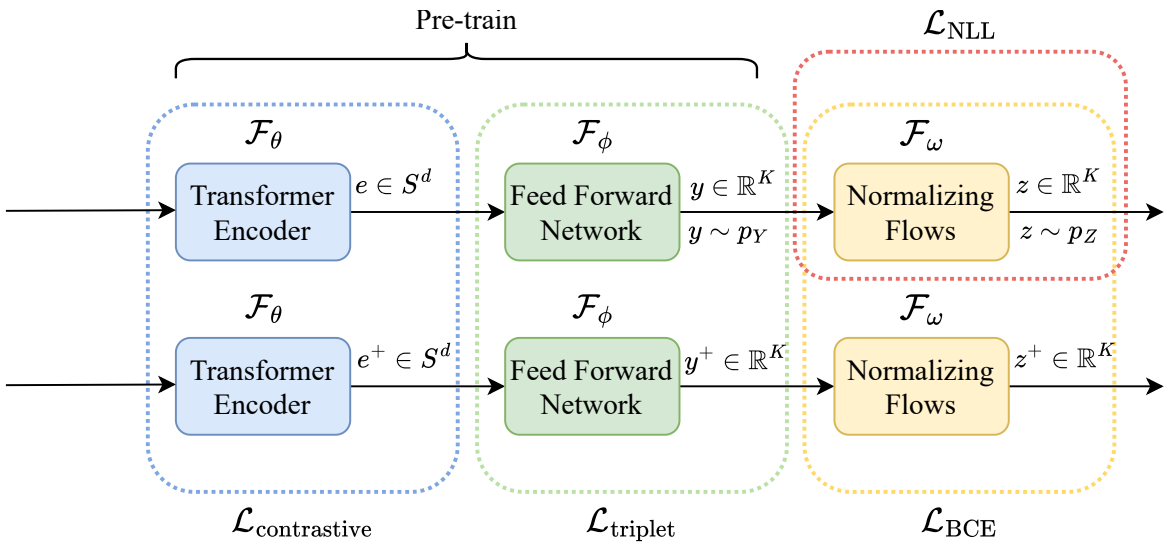

Fig. 1. An overview of the FlowHash model. The encoders $\mathcal{F}_\theta$ and $\mathcal{F}_\phi$ are initially pre-trained in a self-supervised learning framework to generate robust encodings $e \in S^d$ and $y \in \mathbb{R}^K$, respectively, corresponding to the audio segment x. Subsequently, y, combined with a fixed distribution p_Y, is fed into the normalizing flow model $\mathcal{F}_\omega$ to compute $z \in \mathbb{R}^K$. The distribution of z conforms to the desired p_Z, facilitating the generation of balanced hash codes. Additionally, we introduce a regularization term during normalizing flow training to enhance code robustness.

generation [33], leveraging their ability to capture intricate dependencies within the data and generate realistic and high-quality audio samples.

D. Optimal transport

OT has gained traction [34] in machine learning for its ability to quantify the minimal cost of transforming one distribution to another. It finds applications in diverse domains, such as generative modeling [35], domain adaptation [36], and clustering [37]. Notably, Singh *et al.* [9] proposed a novel approach to enhance audio retrieval effectiveness by simultaneously learning audio embeddings and their corresponding binary encodings. They model hashing as a constrained clustering process, necessitating embeddings to be uniformly assigned across all possible K-bit hash codes, a task achieved through leveraging the OT framework. In machine learning, the Sinkhorn-Knopp algorithm [11] solves optimal transport problems efficiently. However, its iterative nature and matrix computations lead to computational complexity, which poses challenges for integration into machine-learning tasks, particularly in large-scale scenarios. Moreover, high memory requirements due to operations on large cost matrices further limit its practicality.

III. Our Approach: FlowHash

Our primary goal is the efficient indexing of the fingerprint database of size N, denoted as $E = \{e_n\}_{n=1}^N$, using a balanced hash table $T = \{h_k : A_k,\ A_k \subset E,\ k \in \{1, 2, ..., 2^K\}\}$. To this end, we first project fingerprints e into a low-dimensional $\mathbb{R}^K$ space while preserving the neighborhood structure. These projections are subsequently transformed using the normalizing flow model to achieve a distribution characterized by 2^K well-balanced modes, each linked to a distinct hash code. This ensures that the size of each A_k is approximately $N/2^K$. We illustrate an overview of our method in Figure 1.

A. Encoder: Representation Learning

We utilize the Transformer-based encoder $\mathcal{F}_\theta$ as initially proposed by [9] to obtain contextualized audio representations. Given a set $\mathcal{D}$ of audio files, we randomly select an audio segment x of fixed length and convert it to a log-Mel spectrogram. We further split the spectrogram into non-overlapping patches along the temporal axis and convert each into a 1D embedding using a projection layer. The sequence of these embedding are then fed into the Transformer encoder. The Transformer output sequence is then concatenated, thereby preserving the temporal structure of the audio. This concatenated sequence is then projected to a d-dimensional embedding, which is subsequently normalized using its L_2 norm. This process results in a *fingerprint* denoted as $e = \mathcal{F}_\theta(x) \in S^d$. Similarly, we generate $e^+ = \mathcal{F}_\theta(x^+)$ for the distorted counterpart x^+ of x. To achieve robust representations, we train the encoder to maximize the cosine similarity between e and e^+ using a contrastive loss:

$$\mathcal{L}_{\text{contrastive}} = -\log \frac{\exp\left((e \cdot e^+)/\tau\right)}{\exp\left((e \cdot e^+)/\tau\right) + \sum_{e^-} \exp\left((e \cdot e^-)/\tau\right)}, \tag{1}$$

where τ is a temperature hyperparameter that facilitates effective learning from hard negatives e^-.

Next, we focus on learning the K-bit hash code h corresponding to each fingerprint e. As a first step, we employ a projection mapping $\mathcal{F}_\phi$, which transforms the fingerprint into a low-dimensional $\mathbb{R}^K$ space, resulting in $y = \mathcal{F}_\phi(e)$. We learn this mapping using the triplet loss to ensure the respective projections of e and e^+ maintain proximity based on the Euclidean distance as:

$$\mathcal{L}_{\text{triplet}} = \max(0, \| y - y^+ \|_2 + \alpha - \| y - \frac{1}{2M-2} \sum_{m=1}^{2M-2} y_m^- \|_2), \tag{2}$$

where α is a margin enforced between positive and negative pairs. Note that in a minibatch comprising M pairs $\{x, x^+\}$, resulting in $2M$ samples, there are $2M$-2 negative samples, x^-, for each pair within the batch. Finally, we pre-train the encoders $\mathcal{F}_\theta$ and $\mathcal{F}_\phi$ using the overall loss that combines both contrastive and triplet losses:

$$\mathcal{L}_{\text{pretrain}} = \mathcal{L}_{\text{contrastive}} + \mathcal{L}_{\text{triplet}} \tag{3}$$

We transform $e \in S^d$ into $y \in \mathbb{R}^K$ to facilitate the subsequent stage of training using NF. A common issue with NF is the occurrence of NaNs in the training loss when the supports of the input and output distributions differ. Therefore, it is essential for stable training with NF to have both distributions with the same support and intrinsic dimension. Normalizing reduces the intrinsic dimension by one. We train the NF to produce a bimodal GMM distribution in each dimension, which exhibits support in the $(-\infty, \infty)$ range. Therefore, it necessitates aligning the input distribution, originally bounded in the L2-normalized embeddings space, with the same support in $\mathbb{R}^K$.

Furthermore, the reason for pre-training the encoders is to establish a fixed distribution p_Y of projections y. This distribution serves as input to the NF model in the next step.

B. Normalizing Flow: Balanced Hashing

We aim to transform samples y into z while ensuring that each component z_k independently follows a bimodal Gaussian mixture distribution p_{Z_k} along its respective k-th dimension. This transformation results in a joint distribution p_Z consisting of 2^K balanced modes in the $\mathbb{R}^K$ space (see Figure 1 in Appendix). To accomplish this, we leverage the RNVP normalizing flow model [15].

The RNVP model applies a series of invertible mappings to transform a simple prior distribution into a complex distribution. Let $Z \in \mathbb{R}^K$ be a random variable with prior distribution p_Z, and $\mathcal{F}_\omega = f_L \circ f_{L-1} \circ ... \circ f_1$ be an invertible function such that $Y = \mathcal{F}_\omega(Z)$. We can determine the probability density of random variable Y using the change of variable of formula as:

$$\log(p_Y(y)) = \log(p_Z(\mathcal{F}_\omega^{-1}(y))) + \sum_{l=1}^{L} \log|\det(J_{f_l^{-1}}(y_l))|, \tag{4}$$

where $J_{f_l^{-1}}$ is the Jacobian of f_l^{-1}. We denote the l-th intermediate flow output as $y_l = f_l \circ f_{l-1} \circ ... \circ f_1(z)$ and thus $y_L = y$. These mappings f_l are parameterized by neural networks, and when applied sequentially, they are able to transform a simple distribution into a more complex one. Moreover, these mappings are designed to be invertible with tractable Jacobian determinants. In RNVP, we define these mappings as affine coupling layers. This layer splits the input $z \in \mathbb{R}^K$ into two disjoint parts, $(z^{(1)}, z^{(2)}) \in \mathbb{R}^k \times \mathbb{R}^{K-k}$ and transforms the input non-linearly as:

$$y^{(1)} = z^{(1)} \tag{5}$$

$$y^{(2)} = z^{(2)} \odot \exp(s(z^{(1)})) + t(z^{(1)}), \tag{6}$$

where $s(.)$ and $t(.)$ are neural networks, and $\odot$ represents the Hadamard product.

Here, we consider z_k of z as mutually independent components, each following a fixed bimodal Gaussian distribution p_{Z_k} and thus we define p_Z as:

$$\log(p_Z(z)) = \sum_{k=1}^{K} \log(p_{Z_k}(z_k)) \tag{7}$$

$$= \sum_{k=1}^{K} \log(0.5 \cdot \mathcal{N}(z_k| - 2, 1) + 0.5 \cdot \mathcal{N}(z_k|2, 1)), \tag{8}$$

where $\mathcal{N}(z; \mu, \sigma^2)$ represents a Gaussian distribution with mean μ and variance σ^2. We train the RNVP model using the forward KL-divergence loss function, which is equivalent to minimizing the negative log-likelihood of the input data:

$$\mathcal{L}_{\text{NLL}} = -\log(p_Y(y)) \tag{9}$$

Here, in contrast to the generative task, we employ the inverse function $\mathcal{F}_\omega^{-1}$ to transform the input embedding y into z. Subsequently, after training the model, we use it to generate a K-bit hash code h by quantizing z as $h = \text{positive}(z)$. Specifically, for each component $k = 1, 2, ..., K$, we define $\text{positive}(z_k) = 1$, if $z_k > 0$, and $\text{positive}(z_k) = 0$ otherwise.

C. Hash codes robustness

We also aim to ensure that both fingerprints in any given pair $\{e, e^+\}$ are consistently mapped to the same hash bucket h. This objective essentially translates to a binary classification problem, where each bit is assigned to either class 0 or 1. Therefore, we introduce cross-entropy based regularization term to achieve robust hash codes. The primary purpose of the regularization is to enforce the embeddings z and z^+ to align closely, thereby promoting the reliable mapping of samples to hash buckets. The regularization term is defined as:

$$\mathcal{L}_{\text{BCE}} = \sum_{k=1}^{K} \big(\mathcal{H}(z_k, z_k^+) + \mathcal{H}(z_k^+, z_k)\big), \tag{10}$$

where $\mathcal{H}(u, v)$ represents the cross-entropy between two distributions u and v. In our context, it quantifies the dissimilarity between z_k and z_k^+. To further clarify, $\mathcal{H}(u, v)$ is computed as:

$$\mathcal{H}(u, v) = -\big[p_{Z_{1k}}(u) \cdot \log(p_{Z_{1k}}(v)) + p_{Z_{2k}}(u) \cdot \log(p_{Z_{2k}}(v))\big], \tag{11}$$

Here, $p_{Z_{1k}}$ and $p_{Z_{2k}}$ denote the probability density associated with two modes of the Gaussian mixture distribution along the k-th dimension. We finally train our model to minimize the overall loss combining negative log-likelihood loss and regularization term:

$$\mathcal{L}_{\text{NF}} = \frac{1}{M} \sum_{m=1}^{M} \big(\mathcal{L}_{\text{NLL}}(z_m) + \lambda \mathcal{L}_{\text{BCE}}(z_m, z_m^+)\big), \tag{12}$$

where the hyperparameter $\lambda \geq 0$ controls the trade-off between balanced hashing and hash code robustness.

D. Indexing

We employ the trained encoders and normalizing flow model to extract fingerprints from a set of reference audio tracks $\mathcal{D}$ and subsequently index them using a hash table T for an efficient retrieval during inference:

- **Fingerprinting**: We first extract audio segments of t seconds from each audio track at a regular interval of $s(<t)$ seconds. Then, we generate a log-Mel spectrogram corresponding to each segment and feed it into the encoder to generate its fingerprint $e \in S^d$.
- **Hashing**: We input the fingerprint e into the projection encoder, followed by the normalizing flow model, which computes $z \in \mathbb{R}^K$. To create the hash code $h \in \{0,1\}^K$, we quantize z using the $\mathrm{positive}(.)$ function. However, this straightforward approach has a limitation - it assigns a bit with full confidence even to a z value near 0. Moreover, their slightly perturbed counterpart z^+ may map to a different bit than that assigned to z, leading to a degradation in retrieval performance. Therefore, we assign both bit 0 and 1 to z values within a small neighborhood $r(>0)$ around 0. As a result, e is mapped to multiple hash codes $h(z) = \{h_1, h_2, \ldots, h_i\}$. This ensures a more robust retrieval performance, mitigating the potential performance loss associated with near-zero z values.
- **Hash Table**: Finally, we build the hash table $T = \big\{h_k : \{e_n \mid h_k \in h(z_n)\}, n = 1, 2, \ldots, N\big\}$, where N is the total number of fingerprints extracted from all reference audio tracks.

E. Retrieval

For a query q of length $\tilde{t}$ $(\geq t)$ seconds, we follow the same steps as above to extract fingerprints $e_q = \{e_{q_o}\}_{o=1}^{O}$, where O is the number of segments extracted from q. For each e_{q_o}, we compute its corresponding hash codes $h(q_o) = \{h_1, \ldots, h_i\}$ depending on a neighborhood $\tilde{r}$. Then, we look up into each hash bucket to retrieve all the bucket candidates, $C = \{T(h_k), h_k \in h(q_o)\}$. To find the best match among the candidates, we perform a linear search as:

$$\underset{p \in C}{\arg\max}\ (e_p \cdot e_{q_o})$$

The parameter $\tilde{r}$ determines the number of hash buckets to probe to collect the best match candidates. A larger $\tilde{r}$ value results in probing more buckets, leading to more exact distance computations. In high distortion settings, z and z^+ are likely to be mapped to different hash buckets. Consequently, it becomes necessary to probe multiple hash buckets, which can be achieved by increasing the $\tilde{r}$ parameter.

As the query is segmented sequentially over time, the retrieved set of best-matching indices must correspond to a contiguous sequence. However, this contiguity is not always achieved due to approximate matching. To this end, we employ a simple yet effective subsequence search strategy. This approach enables us to precisely locate the best-matching subsequence within the reference database that aligns with the query q. Let $i = \{i_1, \ldots, i_O\}$ be the retrieved sequence of indices. Then, we generate all possible sequence candidates S_m, which consists of O consecutive indices, as:

$$S_m = \{i_m + o - m, o = 1, 2, .., O\}, m = 1, 2, \ldots, O \quad (13)$$

Finally, we select the best matching sequence S_{m^*} which has the maximum consensus with i:

$$m^* = \underset{m}{\arg\max}\, |S_m \cap i| \quad (14)$$

IV. Experiments

A. Dataset

Music: We use the Free Music Archive (FMA) [38] as a benchmark dataset prominently used for audio fingerprinting tasks. The dataset comprises three primary subsets: *small*, *medium*, and *large*, each containing 30s audio clips. We use the *small* subset for model training, which consists of 8000 balanced clips representing the top 8 genres in the dataset. This balanced distribution enhances model generalization and mitigates potential biases during model training. We use the *medium* set of 25,000 clips, equivalent to 208 hours, to evaluate systems performance.

Noises: We extracted diverse noise samples from the MUSAN corpus [39], comprising 843 noise clips, for model training. We used a distinct set of noise clips obtained from the ETSI background noise database [40] during the evaluation phase. These clips correspond to various environmental contexts, including babble, cafeteria, car, living room, shopping, train station, and traffic.

RIRs: We used the MUSAN corpus to acquire Room Impulse Responses (RIRs) corresponding to various environmental settings, ranging from small indoor rooms to large rooms such as halls, conference rooms, and churches, totaling 325 RIRs. We used RIRs from the Aachen Impulse Response Database [41] during the evaluation. These RIRs correspond to a t_{60} reverberation time of 0.2 to 0.8s.

B. Metrics

Efficacy is measured using the top-1 recall rate, which indicates the percentage of outcomes where the correct match is found at the top rank. Note that we consider the correct match only if the identified timestamp of a query is within 50ms of the actual timestamp. We define recall rate as:

$$\textit{recall@1} = \frac{\textit{n hits @ top-1}}{\textit{n hits @ top-1} + \textit{n miss @ top-1}} \times 100 \quad (15)$$

Efficiency serves as a quantitative measure of how effectively the database is searched to find the top match for each query sample. Let N denote the total number of database samples and N_p be the total unique points evaluated in the candidates list C , then we define the metric as:

$$\mathrm{scan} = \frac{N_p}{N} \times 100 \quad (16)$$

C. Baselines

We compare our proposed method, FlowHash, against deep learning-based encoding methods introduced by [22] (AE) and [24] (NAFP). Additionally, we employ the recently proposed OT-based method [9] for balanced hashing and the LSH as baseline hash-based methods to demonstrate the effectiveness of our proposed approach for efficient retrieval. Moreover, we compare FlowHash with Product Quantization (PQ) and Inverted File with Product Quantization (IVFPQ). PQ compresses the database and accelerates retrieval through precomputed distance tables for fast approximate nearest neighbor searches. IVFPQ further reduces the search space to improve efficiency. We implemented PQ and IVFPQ using the FAISS library [42] with 32 codebooks, each encoded using 12 bits. In IVFPQ, we used varying numbers of partitions for different test datasets to balance search space and comparison count.

D. Implementation

Augmentation: We randomly apply the following distortions to audio input x to generate x^+ as:

- Noise: We add a randomly selected background noise within a 0-20dB SNR level range.
- Reverberation: We filter the input audio with a randomly chosen RIR to simulate room acoustics.
- Time offset: We add a temporal offset of up to 50ms to account for potential temporal inconsistencies in the real-world scenario.

Training: We jointly train the encoders using Adam optimizer with a learning rate of 1e-4 for 1500 epochs. Specifically, we pre-train the model until the validation accuracy converges. During validation, we perform a query-by-example task on a dummy database using brute force search to assess the effectiveness of the trained model. This dummy database consists of 1000 audio tracks extracted from the *medium* subset of FMA. Further, we train the NF model using the same optimizer with an initial learning rate of 1e-3 for 1200 epochs until the learning rate reaches its minimum value of 1e-4. The training of these models was conducted on a single NVIDIA A100 GPU.

Hyperparameters: We set the margin $\alpha = 0$ because the triplet loss without a margin yields better overall performance in our context. The performance degradation observed with a positive margin can be attributed to the small dimensionality of y embeddings, which limits their capacity to capture complex relationships and increases the likelihood of overlap. For the parameter r, we set $r = 1.0$ to assign bits 0 and 1 to z during indexing of the reference database. A higher r value maps a point to multiple hash buckets, resulting in more exact distance computations. Thus, we selected r to balance efficiency and effectiveness. Our experiments show that approximately 91% of the database samples have up to 4 out of 16 bits assigned as both 0 and 1 when r is 1.
Furthermore, we set $\lambda = 0.5$ to weigh the $\mathcal{L}_{\text{BCE}}$ loss while training the NF model to ensure a balance between achieving uniform hash buckets and generating robust hash codes.

Database: We generate fingerprints for 1s audio segments extracted every 100ms in each audio track, which resulted in a database of ∼7M fingerprints.

Queries: We generate 1,000 queries by randomly selecting segments from the reference audio tracks. These queries vary in length from 1-5s and are distorted with added noise, reverb, or a mix of both. To generate a noisy reverberant audio query, we initially filtered both the original audio and noise signal using RIR corresponding to a t_{60} level of 0.5s. Subsequently, we added the reverberated noise to the filtered audio within the 0-20 dB SNR range.

V. Results

Table I compares our method with baseline approaches, focusing on efficiency and efficacy. We assess effectiveness by comparing the Transformer-based encoding (TE) with NAFP and AE methods, using LSH for indexing and consistent evaluation. TE captures contextual information better, resulting in more discriminative embeddings and consistently achieving 10-20% higher hit rates across varying distortion levels and query lengths. This is particularly beneficial for adjacent audio segments with similar timbre but different temporal evolution. Similar trends are observed in reverberant and noisy conditions, indicating TE's robustness and effectiveness in challenging acoustic environments.

In our experimental observations concerning 1-second queries, notable mismatches arose due to recurring query instances, such as the refrain of a song within an audio track. Consequently, selecting an extended query becomes imperative to be more discriminative and ensure accurate identification of its correct match. This is corroborated by the consistent trend of improved performance across all methods with extended query lengths. Moreover, our subsequence search facilitates precise query alignment within the identified audio track, resulting in a substantial increase in recall rates for longer queries.

We analyze the acceleration achieved by our proposed balanced hashing using NF. We fixed the transformer encoder (TE) to generate the fingerprint database to ensure a fair comparison. We evaluate the retrieval performance based on the proportion of the database considered for exact distance computation with each query fingerprint, as indicated by the "scan" metric (Equation 16) in Table I.
FlowHash demonstrates superior recall rates across most distortion environments. When examining performance under high distortion levels such as 0-10dB SNR, FlowHash consistently outperforms other methods. For instance, FlowHash surpasses LSH by 2.6%, OT by 11.9%, and IVFPQ by 0.2% at 0dB in noisy conditions. Noisy reverberant environments pose more challenges, causing a significant drop in recall rates for all methods due to added non-stationary noises and smearing effects from filtering with an RIR. This leads to numerous mismatches within an identified audio track. IVFPQ's performance drops moderately in these conditions

TABLE I
FMA MEDIUM: COMPARISON OF TOP-1 RECALL RATES (%) IN VARIOUS DISTORTION ENVIRONMENTS FOR VARIED QUERY LENGTHS.

	Method	Noise ↑					Noise + Reverb ↑					Reverb ↑					scan ↓
		0dB	5dB	10dB	15dB	20dB	0dB	5dB	10dB	15dB	20dB	0.2s	0.4s	0.5s	0.7s	0.8s	
1s	NAFP + LSH	50.1	66.4	73.0	75.1	76.0	21.3	43.1	53.9	58.3	60.5	61.4	60.3	57.6	48.5	42.3	2.28
	AE + LSH	59.1	70.2	71.5	74.8	75.7	32.1	52.0	57.9	63.3	65.1	66.2	65.3	63.1	55.3	51.9	2.32
	TE + LSH	74.6	81.9	87.0	89.1	91.4	47.8	67.5	77.8	82.1	83.0	83.4	82.2	80.2	74.0	72.7	2.30
	TE + IVFPQ	77.0	87.8	93.6	93.9	94.7	47.4	68.3	76.1	80.1	80.3	86.0	82.8	81.9	**77.0**	**74.1**	1.52
	TE + OT	65.3	82.0	87.5	89.6	90.2	42.3	62.1	74.1	78.7	81.3	83.5	82.7	80.1	68.0	63.4	1.21
	TE + NF (FlowHash)	**77.2**	**88.0**	**93.9**	**94.1**	**95.8**	**48.3**	**68.3**	**79.8**	**83.1**	**84.4**	**86.2**	**83.0**	**82.2**	75.1	73.1	**1.11**
2s	NAFP + LSH	69.7	79.0	85.0	86.8	87.1	40.1	63.3	72.6	75.0	76.1	77.1	76.7	75.5	68.3	61.0	2.28
	AE + LSH	75.4	83.7	84.5	87.0	87.7	53.1	71.6	78.0	81.6	82.2	84.0	82.5	81.6	75.2	70.8	2.32
	TE + LSH	82.6	90.0	92.3	93.8	94.6	61.5	81.2	85.5	86.3	87.1	90.0	88.6	87.0	80.6	76.9	2.30
	TE + OT	81.6	90.0	92.6	93.6	94.2	59.2	80.6	85.1	86.1	87.3	91.6	88.5	86.0	79.1	71.1	1.21
	TE + NF (FlowHash)	**89.0**	**94.2**	**95.6**	**96.4**	**97.0**	**62.4**	**83.4**	**87.0**	**88.0**	**88.3**	**92.1**	**90.3**	**88.9**	**82.4**	**78.5**	**1.12**
3s	NAFP + LSH	77.0	83.8	88.0	88.7	89.1	53.6	71.1	77.7	78.5	81.0	82.3	78.6	76.5	69.1	62.3	2.28
	AE + LSH	81.7	86.5	88.1	88.7	89.4	64.4	78.1	82.4	85.0	86.7	88.8	85.6	84.0	76.7	72.3	2.32
	TE + LSH	86.5	92.2	94.1	94.9	96.0	70.6	85.5	88.7	88.6	89.3	92.8	89.6	89.0	81.3	78.2	2.30
	TE + OT	85.0	91.1	95.0	96.0	96.6	70.0	83.8	88.4	88.7	89.4	93.1	91.0	87.2	79.5	73.1	1.21
	TE + NF (FlowHash)	**91.7**	**95.6**	**96.1**	**97.0**	**97.1**	**72.7**	**87.8**	**89.1**	**89.5**	**90.1**	**94.0**	**91.7**	**90.7**	**83.7**	**79.1**	**1.11**
5s	NAFP + LSH	81.4	88.2	90.3	91.5	91.6	62.1	80.1	82.5	82.7	83.6	84.5	83.1	78.1	74.0	64.8	2.28
	AE + LSH	83.1	89.0	90.8	91.2	92.0	76.3	85.1	87.1	89.0	89.1	90.5	88.0	86.3	80.0	76.9	2.32
	TE + LSH	89.9	92.4	95.4	96.4	96.5	81.3	90.0	90.2	90.9	90.7	94.0	91.2	90.1	82.3	79.0	2.30
	TE + OT	88.1	93.4	95.3	96.1	97.0	80.1	88.2	90.8	90.8	91.0	95.2	91.4	88.3	79.4	74.9	1.21
	TE + NF (FlowHash)	**93.8**	**96.6**	**97.0**	**97.3**	**97.4**	**82.0**	**90.8**	**90.8**	**91.2**	**91.4**	**95.6**	**92.7**	**92.4**	**84.5**	**80.9**	**1.11**

TABLE II
FMA MEDIUM: COMPARISON OF TOP-1 RECALL RATES(%) WITH EXHAUSTIVE SEARCH METHODS IN VARIOUS DISTORTION ENVIRONMENTS FOR VARIED QUERY LENGTHS.

	Method	Noise ↑						Noise + Reverb ↑						Reverb ↑					scan ↓
		-5dB	0dB	5dB	10dB	15dB	20dB	-5dB	0dB	5dB	10dB	15dB	20dB	0.2s	0.4s	0.5s	0.7s	0.8s	
1s	TE + Brute Force	58.7	83.7	91.8	94.3	94.8	96.4	27.1	61.1	82.7	88.2	90.7	91.3	87.0	83.7	82.8	80.0	76.0	100
	TE + PQ	57.7	82.8	91.3	93.9	94.5	96.3	25.9	60.2	81.5	87.2	90.2	91.0	86.6	83.3	82.2	78.7	74.6	100
	TE + NF (FlowHash)	58.0	83.1	91.5	94.1	94.8	96.5	25.5	59.7	81.9	87.6	90.2	91.1	86.8	83.4	82.4	79.2	74.9	2.53

due to quantization, which reduces the discriminative power of compressed fingerprints needed for precise query timestamp locations. In reverb-only conditions, IVFPQ performs competitively with FlowHash but requires more distance computations, highlighting FlowHash's efficiency and robustness.

Similar trends are observed for longer queries where FlowHash consistently outperforms the baselines. Notably, the performance gap between the methods decreases with increasing lengths, particularly in low distortion conditions, but this comes at the cost of a large search space compared to our method.

FlowHash excels in retrieval efficiency, requiring the evaluation of only 1.11% of the database, compared to 1.52% and 2.30% for IVFPQ and LSH, respectively. It also achieves higher recall rates than OT while evaluating a similar percentage of samples. By generating well-separated hash buckets, FlowHash reduces the likelihood of anchor and positive samples mapping to different buckets at high distortion levels, thereby contributing to higher recall rates while maintaining evaluation percentages similar to OT. It is important to highlight that, unlike OT, which evaluates the similarity of a query with all 2^K hash buckets, adding significant overhead, our method probes fewer buckets based on near zero-valued z-values, making it computationally efficient. On average, our method probes 94 buckets, compared to 1000 for OT and LSH.

In Table II, we compare top-1 recall rates across different distortion environments for 1-second queries using Brute Force (BF), Product Quantization (PQ), and our proposed FlowHash method under various distortion levels, including challenging negative SNR levels at -5dB. The BF method sets the benchmark for recall performance, achieving the highest recall rates across all distortion environments due to its exhaustive search strategy. We observe a drop of up to 1.4% for the PQ method across different distortion environments. Our method competes favourably with exhaustive search methods, given that it only evaluates 2.53% of the reference database. However, under the combined condition of -5dB noise and reverb, all methods experience a significant performance drop, indicating the increased difficulty of this scenario.

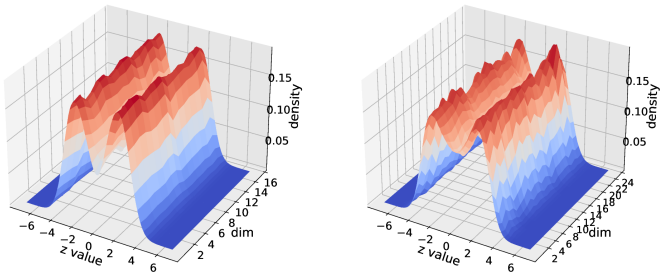

Fig. 2. The illustration shows the marginal distributions, p_{Z_k}, for each dimension. Each dimension exhibits a bimodal Gaussian mixture distribution for K=16 (left) and K=24 (right). This indicates that the joint distribution p_Z consists of well-balanced 2^K modes in the $\mathbb{R}^K$ space.

VI. DISCUSSION

A. Failure cases

Our analysis identified duplicate audio tracks in the reference database, which caused queries to match with these

This article has been accepted for publication in IEEE/ACM Transactions on Audio, Speech and Language Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2024.3486227

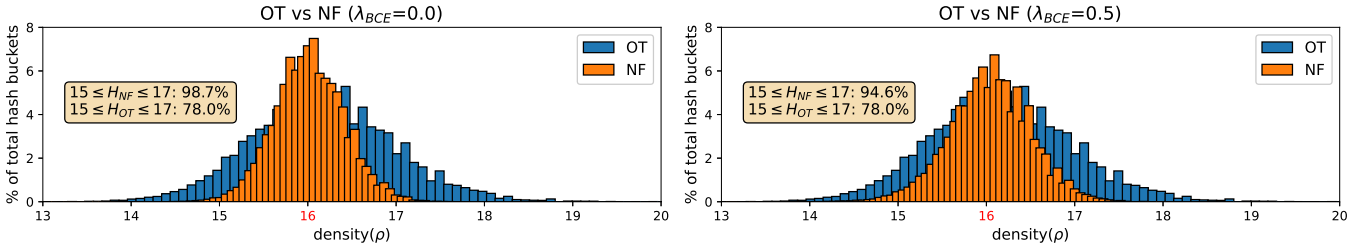

Fig. 3. Comparison of hash code balance between NF and OT for K=16.

duplicates instead of the intended tracks, resulting in a 1.1% drop in recall rates across all distortion settings. Furthermore, while 1-second queries corresponding to the chorus of a song were correctly identified, they often matched with a chorus located at a different timestamp within the same song, contributing further to the decrease in recall rates. Moreover, in scenarios with high distortion levels, such as 0-5dB SNR, the NF mappings (i.e., z and z^+) were found to be distant and located in different GMM modes. This discrepancy led to missing the appropriate hash bucket to probe during inference, thereby failing to retrieve the best match.

B. Target distribution

In Figure 2, we present an analysis for K=16, demonstrating that the normalizing flow effectively yields the target distribution p_Z. In particular, we analyze the marginal distributions p_{Z_k} of p_Z along each dimension. These marginals are depicted as a bimodal Gaussian mixture with balanced modes, resulting in an overall balance across all 2^{16} modes in the $\mathbb{R}^{16}$ space. Notably, we achieve this balance mode distribution even for a larger K, such as K=24. This highlights the applicability of our approach in scenarios where a substantial number of hash buckets are required to index an extensive database efficiently.

C. Balanced hash codes

We evaluate the balance of K-bit hash codes by examining the density of each hash bucket within a hash table T. The density measures the proportion of total samples N mapped to a hash bucket h_k, and is defined as $\rho_k = -\log_2(|T(h_k)|/N)$. A uniform distribution of density values indicates an optimal balance, each attaining the value of K. This signifies an ideal scenario where samples are evenly distributed across all possible 2^K hash buckets. Deviations in density from K, either higher or lower, indicate that the respective bucket is either underfilled or overfilled. We show in Figure 3 that NF achieves 16-bit hash codes with a more balanced distribution compared to the OT formulation. Our method results in ∼95% of total hash buckets whose density lie between 15 and 17, as opposed to ∼78% in the OT-based approach. Adding the $\mathcal{L}_{\text{BCE}}$ loss introduces some disruption to this balance; however, it still outperforms the OT method.

D. Regularization loss

To assess the effectiveness of the regularization term, we use the test database to analyze the absolute value difference between z_k and z_k^+ across all K dimensions. Figure 4 shows that z_k and z_k^+ tend to lie closer when employing the $\mathcal{L}_{\text{BCE}}$ loss function during training. Consequently, this increases the probability of z_k and z_k^+ being assigned to the same bit and thus requires fewer bucket probes during the search. On the contrary, a substantial difference in z values is observed without $\mathcal{L}_{\text{BCE}}$ loss. Our analysis indicates that ∼82% of all pairs (z, z^+) exhibit less than a 0.5 z-value difference, compared to only ∼51% in the absence of $\mathcal{L}_{\text{BCE}}$.

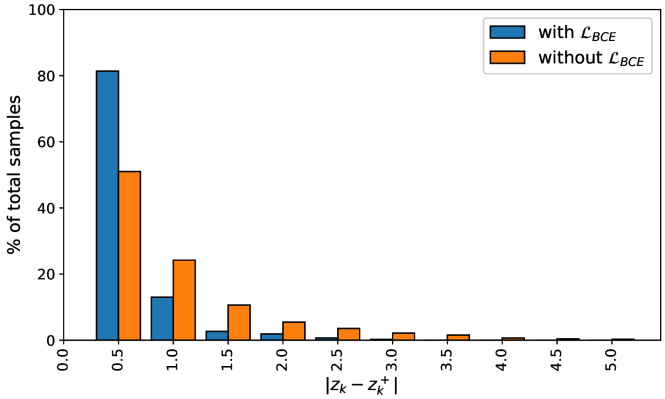

Fig. 4. Effect of $\mathcal{L}_{\text{BCE}}$ on the absolute difference between z_k and z_k^+. The addition of the $\mathcal{L}_{\text{BCE}}$ loss coaxes z_k and z_k^+ to map them to the same bit, whereas its absence results in an increased distance between them.

E. Scalability

In Table III, we compare the performance of our method using an industrial dataset of broadcast content. This dataset includes various content types such as awards shows, folk music, sports programs, autobiographies, and interviews. The dataset exhibits several audio artifacts, including reverberation, compression artifacts, and inconsistent volume levels, reflecting the diversity and complexity of real-world broadcasts. We trained our model on a subset of approximately 500 hours of data and used the remaining dataset, totalling around 1950 hours, to construct a fingerprint database for evaluation. Our reference database consists of ∼74M fingerprints. We trained our model to output 24-bit hash codes to accommodate the large database size. Notably, we did not include results for OT due to computational constraints in computing 24-bit hash codes.

The results presented in Table III illustrate the competitive performance of our proposed method compared to baselines.

TABLE III
BROADCAST CONTENT: COMPARISON OF TOP-1 RECALL RATES (%) IN VARIOUS DISTORTION ENVIRONMENTS FOR 1 SECOND QUERIES. WE UNDERLINE OUR RESULTS IF THE ACCURACY DROP IS LESS THAN 1.5% COMPARED TO THE BEST PERFORMING BASELINE.

	Method	Noise ↑					Noise + Reverb ↑					Reverb ↑					scan ↓
		0dB	5dB	10dB	15dB	20dB	0dB	5dB	10dB	15dB	20dB	0.2s	0.4s	0.5s	0.7s	0.8s	
1s	TE + LSH	**52.1**	61.7	71.4	82.5	85.8	33.2	**52.5**	64.6	75.0	80.0	83.8	81.6	**80.0**	**75.3**	**68.4**	0.39
	TE + IVFPQ	51.7	61.3	72.0	**83.7**	**86.2**	32.1	52.0	64.1	76.3	81.7	84.3	82.0	79.0	74.7	67.3	0.28
	TE + NF (FlowHash)	51.3	**62.5**	**72.5**	83.4	86.0	**34.0**	51.9	**65.1**	**77.1**	**82.3**	**84.4**	**82.0**	78.9	74.1	67.0	**0.22**

TABLE IV
THE EFFECT OF WIDTHS r AND $\tilde{r}$ AND THE FINGERPRINTS DIMENSION d AT DIFFERENT DISTORTION LEVELS ON EFFICACY AND EFFICIENCY FOR 1SEC QUERIES.

Noise (dB)																				
	$\mathbf{r=0.5,\ \tilde{r}=1.0}$					$\mathbf{r=0.5,\ \tilde{r}=1.5}$					$\mathbf{r=1.0,\ \tilde{r}=1.0}$					$\mathbf{r=1.0,\ \tilde{r}=1.5}$				
	0	5	10	15	scan	0	5	10	15	scan	0	5	10	15	scan	0	5	10	15	scan
d=128	45.3	66.1	80.0	87.0	0.12	60.1	76.3	88.6	92.7	0.57	54.5	76.0	87.1	91.4	0.37	77.2	88.0	93.9	94.1	1.12
d=256	46.5	68.1	81.3	87.2	0.11	64.1	78.1	89.2	91.6	0.53	55.8	76.9	87.4	91.5	0.34	77.7	89.4	94.0	95.0	1.10
d=512	48.3	68.8	83.5	89.0	0.12	64.6	79.1	91.6	92.3	0.56	58.0	77.3	88.7	92.0	0.34	78.1	90.3	94.2	95.0	1.20
Noise + Reverb (dB)																				
	$\mathbf{r=0.5,\ \tilde{r}=1.0}$					$\mathbf{r=0.5,\ \tilde{r}=1.5}$					$\mathbf{r=1.0,\ \tilde{r}=1.0}$					$\mathbf{r=1.0,\ \tilde{r}=1.5}$				
	0	5	10	15	scan	0	5	10	15	scan	0	5	10	15	scan	0	5	10	15	scan
d=128	20.0	37.0	54.3	58.4	0.12	30.3	53.8	71.7	74.8	0.54	30.0	52.1	67.1	72.5	0.35	48.3	68.3	79.8	83.1	1.11
d=256	20.0	37.1	54.0	58.3	0.10	33.3	57.4	70.6	76.4	0.48	30.4	52.0	67.4	73.4	0.32	48.1	68.2	79.4	84.1	1.00
d=512	22.1	43.3	55.6	62.7	0.12	35.1	58.0	74.3	78.0	0.57	33.9	56.8	70.0	76.5	0.37	48.6	71.6	81.3	86.4	1.21

Specifically, our method achieves comparable, and in some cases higher, recall rates with fewer comparisons required across various distortion environments. Moreover, the critical advantage of our approach lies in its efficiency. On average, our method requires exact distance computation with only 0.22% of the total database, compared to 0.39% for LSH and 0.28% for IVFPQ. This indicates that FlowHash is $1.77\times$ and $1.28\times$ more efficient than LSH and IVFPQ while achieving competitive recall rates. This efficiency is particularly notable at higher distortion levels, where maintaining performance with fewer comparisons is challenging.

F. Ablation

The ablation study investigates the effect of varying the widths r and $\tilde{r}$ and the fingerprint dimensions d on the recall rates and evaluation metric across different SNR levels in both noisy and noisy reverb conditions for 1-second queries. We observe from Table IV that increasing values for r and $\tilde{r}$ result in higher recall rates across all the conditions. However, this also entails a higher number of candidate evaluations because increasing r leads to an increase in hash bucket size and increasing $\tilde{r}$ requires probing more buckets during the search. Therefore, to balance accuracy and the total candidate evaluation, we opted for $r = 1.0$ and $\tilde{r} = 1.5$ during the evaluation. Furthermore, the results indicate that increasing the fingerprint dimension further improves the recall rates.

VII. LIMITATIONS

The main limitation of the proposed method lies in the computational complexity associated with generating balanced hash codes using NF. However, the balanced K-bit hash codes speed up the retrieval process, particularly as K increases. Additionally, our method utilizes a computationally intensive encoder to generate fingerprints, enabling precise matching of queries even in high-distortion environments. While these choices do come with a trade-off in terms of longer encoding time, they ultimately contribute to the great efficacy and retrieval efficiency of our method. In terms of memory requirements, each fingerprint is encoded using 4 bytes, occupying a total of 4.8GB for $\sim$7M fingerprints. However, this space requirement could be reduced by half to 2.4GB by encoding each fingerprint using 2 bytes with only a negligible ($\leq 0.5\%$) decline in the accuracy.

VIII. CONCLUSION AND FUTURE WORK

This paper presents a novel application of normalizing flows (NF) in the vector search domain, particularly for the audio fingerprinting task. We leverage NF to generate balanced K-bit hash codes, consequently facilitating efficient database indexing by constructing a balanced hash table. We employ a self-supervised learning framework to enhance the robustness of our system against high noise and reverberation levels. Additionally, we incorporate a regularization loss while training NF to ensure consistent mapping of vectors and their corresponding noisy variants to the same hash bucket, thereby strengthening the indexing process. Our empirical findings demonstrate that the proposed balanced hashing approach accelerates retrieval concerning the number of exact discount computations without compromising the recall rates compared to the baselines. Moreover, our system exhibits scalability and efficiency in retrieval, outperforming the baseline approaches. Future work will focus on optimizing our binary hashing method and exploring hybrid approaches that combine the strengths of hashing and quantization techniques.

IX. ACKNOWLEDGMENT

This work has been supported by the research grant (PB/EE/2021128-B) from Prasar Bharati. We would also like to extend our gratitude to CDAC-Paramsiddhi for their GPU computing resources.

REFERENCES

[1] X. Luo, H. Wang, D. Wu, C. Chen, M. Deng, J. Huang, and X.-S. Hua, "A survey on deep hashing methods," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, pp. 1–50, 2023.

[2] A. Wang *et al.*, "An industrial strength audio search algorithm." in *Ismir*, vol. 2003, 2003, pp. 7–13.

[3] P. Saadatpanah, A. Shafahi, and T. Goldstein, "Adversarial attacks on copyright detection systems," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 8307–8315.

[4] V. Lohmüller and C. Wolff, "Towards a comprehensive definition of second screen," *Proceedings of Mensch und Computer 2019*, 2019.

[5] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, pp. 117–128, 2010.

[6] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, pp. 2916–2929, 2012.

[7] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, 1999, pp. 518–529.

[8] J. Gao, H. V. Jagadish, W. Lu, and B. C. Ooi, "Dsh: data sensitive hashing for high-dimensional k-nnsearch," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1127–1138.

[9] A. Singh, K. Demuynck, and V. Arora, "Simultaneously learning robust audio embeddings and balanced hash codes for query-by-example," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[10] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[11] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, 2013.

[12] X. Zheng, Y. Zhang, and X. Lu, "Deep balanced discrete hashing for image retrieval," *Neurocomputing*, vol. 403, pp. 224–236, 2020.

[13] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, pp. 437–451, 2017.

[14] J. T. Hoe, K. W. Ng, T. Zhang, C. S. Chan, Y.-Z. Song, and T. Xiang, "One loss for all: Deep hashing with a single cosine similarity based learning objective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 286–24 298, 2021.

[15] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[16] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system." in *Ismir*, vol. 2002, 2002, pp. 107–115.

[17] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 597–604.

[18] B. Gfeller, R. Guo, K. Kilgour, S. Kumar, J. Lyon, J. Odell, M. Ritter, D. Roblek, M. Sharifi, M. Velimirović *et al.*, "Now playing: Continuous low-power music recognition," *arXiv preprint arXiv:1711.10958*, 2017.

[19] A. Báez-Suárez, N. Shah, J. A. Nolazco-Flores, S.-H. S. Huang, O. Gnawali, and W. Shi, "Samaf: Sequence-to-sequence autoencoder model for audio fingerprinting," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, pp. 1–23, 2020.

[20] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern recognition*, vol. 41, pp. 3467–3480, 2008.

[21] X. Wu and H. Wang, "Asymmetric contrastive learning for audio fingerprinting," *IEEE Signal Processing Letters*, vol. 29, pp. 1873–1877, 2022.

[22] A. Singh, K. Demuynck, and V. Arora, "Attention-based audio embeddings for query-by-example," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022*, 2022, pp. 52–58.

[23] T. Shibuya, M. Abe, and M. Nishiguchi, "Audio fingerprinting robust against reverberation and noise based on quantification of sinusoidality," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–6.

[24] S. Chang, D. Lee, J. Park, H. Lim, K. Lee, K. Ko, and Y. Han, "Neural audio fingerprint for high-specific audio retrieval based on contrastive learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3025–3029.

[25] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: efficient indexing for high-dimensional similarity search," in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 950–961.

[26] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal lsh for angular distance," *Advances in neural information processing systems*, vol. 28, 2015.

[27] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and efficiency in high dimensional nearest neighbor search," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 563–576.

[28] R. Panigrahy, "Entropy based nearest neighbor search in high dimensions," *arXiv preprint cs/0510019*, 2005.

[29] V. Satuluri and S. Parthasarathy, "Bayesian locality sensitive hashing for fast similarity search," *arXiv preprint arXiv:1110.1328*, 2011.

[30] Y. Dong, P. Indyk, I. Razenshteyn, and T. Wagner, "Learning space partitions for nearest neighbor search," *arXiv preprint arXiv:1901.08544*, 2019.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[33] P. Esling, N. Masuda, A. Bardet, R. Despres *et al.*, "Universal audio synthesizer control with normalizing flows," *arXiv preprint arXiv:1907.00971*, 2019.

[34] L. C. Torres, L. M. Pereira, and M. H. Amini, "A survey on optimal transport for machine learning: Theory and applications," *arXiv preprint arXiv:2106.01963*, 2021.

[35] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[36] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," *Advances in neural information processing systems*, vol. 30, 2017.

[37] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in neural information processing systems*, vol. 33, pp. 9912–9924, 2020.

[38] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.

[39] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[40] S. Processing, "Transmission and quality aspects (stq); speech quality performance in the presence of background noise; part 1: Background noise simulation technique and background noise database," *ETSI EG*, vol. 202, pp. 396–1, 2008.

[41] M. Jeub, M. Schafer, and P. 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.

[42] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," *arXiv preprint arXiv:2401.08281*, 2024.