International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 1, February 2021, pp. 879~891
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp879-891

Similarity-preserving hash for content-based audio retrieval using unsupervised deep neural networks

Petcharat Panyapanuwat, Suwatchai Kamonsantiroj, Luepol Pipanmaekaporn
Department of Computer and Information Science, King Mongkut's University of Technology North Bangkok, Thailand

Article Info

Article history:

Received Jan 1, 2020
Revised Jun 8, 2020
Accepted Aug 18, 2020

Keywords:

Content-based audio retrieval
Deep learning
Deep neural networks
Similarity-preserving hash
Unsupervised learning

ABSTRACT

Due to its efficiency in storage and search speed, binary hashing has become an attractive approach for a large audio database search. However, most existing hashing-based methods focus on data-independent scheme where random linear projections or some arithmetic expression are used to construct hash functions. Hence, the binary codes do not preserve the similarity and may degrade the search performance. In this paper, an unsupervised similarity-preserving hashing method for content-based audio retrieval is proposed. Different from data-independent hashing methods, we develop a deep network to learn compact binary codes from multiple hierarchical layers of nonlinear and linear transformations such that the similarity between samples is preserved. The independence and balance properties are included and optimized in the objective function to improve the codes. Experimental results on the Extended Ballroom dataset with 8 genres of 3,000 musical excerpts show that our proposed method significantly outperforms state-of-the-art data-independent method in both effectiveness and efficiency.

Corresponding Author:

Petcharat Panyapanuwat,
Department of Computer and Information Science,
King Mongkut's University of Technology North Bangkok,
Bangkok, Thailand.
Email: panyapetch@hotmail.com

1. INTRODUCTION

With rapidly growing database of digital audio recordings, the novel retrieval strategies have received great attention. Early retrieval approach uses textual metadata describing the content of music audio (e.g., artist name, song title, album name, genre, or release year of music). In case such descriptions are not available, it is required content-based retrieval strategy that the perceptual aspects of the audio are utilized. [1].

Content-based audio retrieval approach is generally solved with two steps: first, features are extracted from the audio file and then used to build indexes for searching. Two main issues of performing a search over a large database are search speed and efficient storage. The most interesting approach for handling these problems is binary hashing, where the high-dimensional features are encoded into compact binary codes.

There have been several hashing methods proposed in the literature. They can be devided into two categories, data-independent methods and data-dependent methods. Methods in data-independent category [2-7] use random linear projections or some arithmetic expression to construct hash functions. Without the training process, they are robust to data variation. However, such methods require long hash codes to achieve high precision. This increases the storage cost and degrades the search efficiency [8].

Methods in data-dependent category, also called learning to hash methods, aim to learn a set of hash functions from available training data that yield compact codes to achieve satisfactory search performance [9]. Existing data-dependent methods can be classified into unsupervised, supervised, and semi-supervised

learning approach. Unsupervised hashing methods [10-12] use unlabeled data to build the hash functions where the neighbor distance (e.g., L2 norm) among the training data is preserved. Supervised or semi-supervised hashing methods [13-17] attemp to improve the quality of hashing by leveraging the semantic labels into the learning process. Compared with data-independent methods, it appears that data-dependent methods can achieve better accuracy with shorter codes [12, 14, 17]. However, data-dependent methods may be too dependent on the training data [18].

There are both advantages and shortcomings of using data-independent and data-dependent methods. However, the previous works of the two categories do not fully take into consideration the similarity preserving and this may degrade the retrieval performance. In this work, an unsupervised similarity-preserving hashing method for content-based audio retrieval is proposed. We develop a deep network with several hierarchical layers of nonlinear and linear transformations to learn compact binary codes where the similarity between samples is preserved. Furthermore, the independence and balance properties are included in the objective function to improve the codes. The proposed method is compared with the Shazam algorithm [3], the data-independent hashing method, in terms of accuracy, precision, recall, false positive rate, and the storage cost.

2. BACKGROUND

2.1. Learning to hash

Learning to hash attempts to learn a hash function $y = h(x)$ that maps a high-dimensional input item $x \in R^D$ to a compact code y, aiming to improve the search performance [19]. There are 4 topics to consider for the learning to hash: (1) hash function, (2) similarity-preserving, (3) loss function, and (4) deep learning to hash.

2.1.1. Hash function

There are several ways to design hash functions. The most widely used hash functions are generalized by linear projection as shown in (1).

$$y = h(X) = \text{sgn}(f(W^T X + b)) \quad (1)$$

where $y \in \{0,1\}$ or $\{-1,1\}$, $X = \{\text{x}_n\}_{n=1}^N \in R^{D\text{x}N}$ is the training set which contains N samples, D is the dimension of input vector, $W = \{w_k\}_{k=1}^K \in R^{D\text{x}K}$ is the projection vector, K is number of hash bits, b is the bias variable, $\text{sgn}(z) = -1$ or 0 if $z < 0$ and $\text{sgn}(z) = 1$ otherwise, $f(\cdot)$ is a predefined function which can possibly be neural networks or nonlinear function. However, using different $f(\cdot)$ yields different hash function properties.

2.1.2. Similarity-preserving

The distance d_{ij} between two items x_i and x_j can be defined by the standardized Euclidean distance $\|x_i - x_j\|_2$ or others. The similarity s_{ij} between those items is often defined as a function of the distance d_{ij} (e.g., Gaussian function, cosine similarity, and so on). In addition, the semantic similarity approach is generally used in similarity search application. We can apply any distance to the hashing algorithm for semantic similarity, such as Euclidean distance, by defining semantic similarity $s_{ij} = 1$ for adjacent points and $s_{ij} = 0$ or -1 for farther points.

In the hash coding space, the Hamming distance d_{ij}^H between the code y_i and y_j can be defined as $\|y_i - y_j\|_1 = \sum_{k=1}^K \|h_k(x_i) - h_k(x_j)\|$. It is the number of binary digits where the values are different. Hamming similarity is defined as $s_{ij}^H = K - d_{ij}^H$ for the codes valued by 1 and 0. For the codes valued by 1 and -1, the inner product $s_{ij}^H = {y_i}^T y_j$ is defined as the similarity.

Let's focus on the term of similarity preserving. In Figure 1(a), there is a set of three points (x_1, x_2, and x_3) in an input space. By measuring the Euclidean distance between the points, we can find that x_1 is closer to x_2 than to x_3, i.e., x_1 is more similar to x_2 than x_3. The $h(x_1)$, $h(x_2)$, and $h(x_3)$ are the representations of x_1, x_2, and x_3 in the hash coding space (or Hamming space), respectively. From the Figure 1(b), we can see $h(x_1)$ is closer to $h(x_3)$ while $h(x_2)$ is far away. In this case, it shows that the similarities are not preserved. Figure 1(c), on the other hand, shows an example of the similarities that are well preserved.

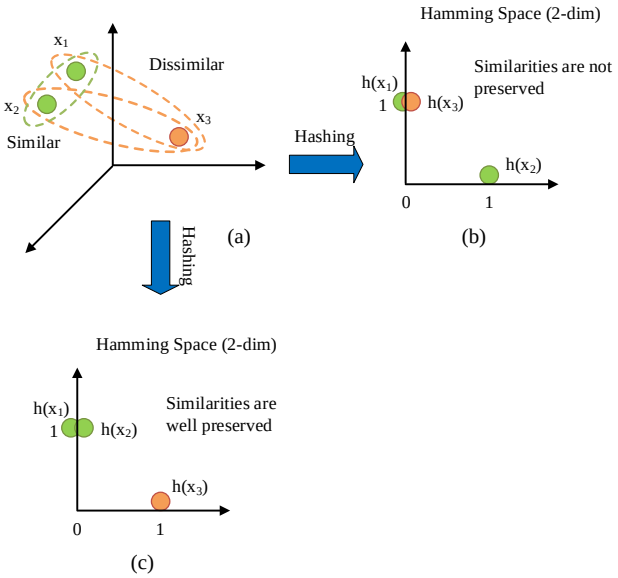

Figure 1. Similarity-preserving hashing

2.1.3. Loss function

The loss function is intended to preserve the similarity order, i.e., minimize the difference between the nearest neighbor search result in the hash coding space and the search result in the input space. The loss function $Loss(X, W)$ is defined as follows:

$$Loss(X, W) = argmin \sum_{x_i, x_j \in X}^{N} \left\| d_{ij} - d_{ij}^{H} \right\|_2 \quad (2)$$

where X is the input data, and W is the projection vector.

Specifically, $y_i = h(x_i)$ needs to be binary. This binary constraint leads to a difficult optimization problem. To solve the problem, we drop the binary constraint and let the codes be continuous. The codes are then binarized with thresholding. For binary constraint relaxation, various standard optimization techniques can be applied.

2.1.4. Deep learning to hash

The goal of learning to hash is to learn the specific hash functions that map high dimensional input vector to a compact binary vector that yields a good quality of retrieval and search speed [20]. For unlabeled data, an illustration of unsupervised deep learning to hash model that map the input vector $x \in R^D$ to compact binary codes is shown in Figure 2.

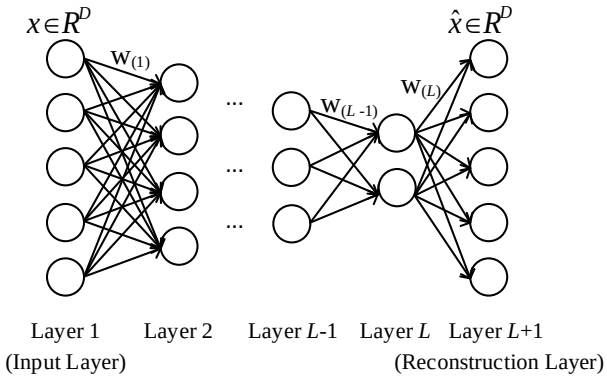

Figure 2. Unsupervised deep learning model

Assume that an unsupervised deep network consists of L+1 layers. A binary vectoy y_i is generated by passing the input vector x_i through the network that contains multiple hierarchical layers of nonlinear functions. The binary code of x_i at Lth layer can be calculated as follows:

$$y_i = h(x_i) = sgn(F(x_i, W)) \quad (3)$$

where $F(x_i, W)$ is a composition of nonlinear transformations defined as follows:

$$F(x_i, W) = f_L(\cdots f_2(f_1(x_i, w_{(1)}), w_{(2)}) \cdots w_{(L)}) \quad (4)$$

where the vector x_i and the weight vector $w_{(l)}$ are used as input, the projection x_{i+1} is produced by $f_i(\cdot)$. The learning algorithm aims to learn a set of nonlinear weight vectors $W = \{w_{(1)}, \ldots, w_{(L)}\}$ where the information from the input space is preserved.

2.2. Search with hashing

There are two strategies to perform a search with hashing, hash code ranking and hash table lookup [19]. For the hash code ranking, an exhaustive search is performed by comparing the distance (e.g., Hamming distance) between the query and the reference items. The items with the smallest distances, called nearest neighbors, are retrieved. However, the cost of computing the distance results in performance degradation. The alternative approach, hash table lookup aims to accelerate the search by reducing the distance computations. The inverse lookup database, called hash table, is composed of buckets which are indexed by the hash codes. Given the query, the matching items storing in the bucket are retrieved.

2.3. Audio fingerprinting

Audio fingerprinting is best known for its ability to identify an uknown audio recording by using its compact content-based signature so-called fingerprint [21]. It does this by converting the audio features into hash codes, aiming to uniquely identify an audio recording. The advantage of fingerprint is that, it reduces storage costs as fingerprint is relatively small. Moreover, the perceptual irrelevancies have been removed from fingerprint, resulting in efficient comparison and searching.

3. METHOD

The aim of this paper is to provide an efficient technique that yields a good quality of retrieval and computational efficiency. In this work, compact binary codes are learned for fingerprint indexing with unsupervised deep network in a way that the similarity between samples is preserved. Once, a short audio sample is taken to our content-based audio retrieval system, the system performs database lookup for matching track and then returns the song ID that the query is taken. As shown in Figure 3, the system is designed with three steps: (1) Fingerprint feature extraction, (2) Unsupervised similarity-preserving hashing, and (3) Sequence matching.

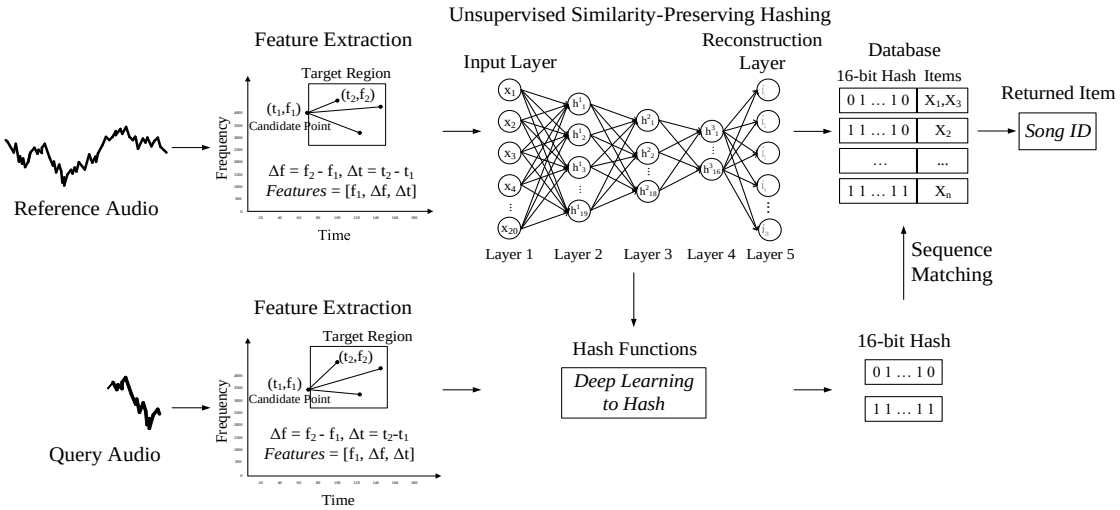

Figure 3. The construction of proposed method for our content-based audio retrieval system

3.1. Fingerprint feature extraction

Before fingerprint feature extraction is performed, the audio signal is converted into a common format for analysis. Next, the time-series audio signal is converted into time-frequency domain from which more meaningful information can be extracted. Below each aspect is detailed.

3.1.1. Preprocessing and transform

In this paper, the fingerprint extraction presented in [3] is applied. Firstly, we convert the input audio to mono signal and downsampled from the standard digital audio of 44.1KHz, to 8KHz, to make the data easier to handle, reducing database size, and increasing speed of the algorithm. The audio signal is then converted into the time-frequency representation. We perform a short-time fourier transform (STFT) with a window size of 64 ms. for good spectrum resolution [22] and a hop size of 32 ms. Figure 4 shows time-frequency graph so-called a spectrogram. On the horizontal axis is time, on the vertical is frequency, and on the third is intensity. Each point on the graph represents the intensity of a given frequency at specific time.

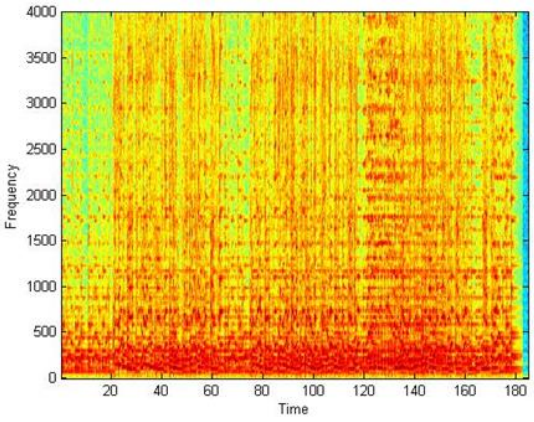

Figure 4. Spectrogram with peak intensities

3.1.2. Feature extraction

After converting the signal into the time-frequency domain, the features are then extracted from the spectrum. Due to their robustness to noise and distortions, the amplitude peaks in each frame are selected as candidate points. Each candidate point is paired with the adjacent peaks. The constellation map of paired points with coordinate list is shown in Figure 5. In this work, each candidate point is paired within 31 frequency bins and 63 time frames. Only the closest 3 peaks in time to each other are selected. Figure 6 shows the combinatorial association of a pair of two points which is called a 'landmark'. For each pair, it consists of four components, the starting frequency f_1, the starting time t_1, the end frequency f_2, and the end time t_2.

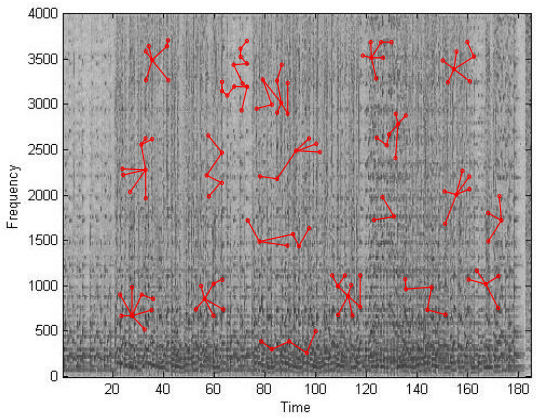

Figure 5. A constellation map of paired points

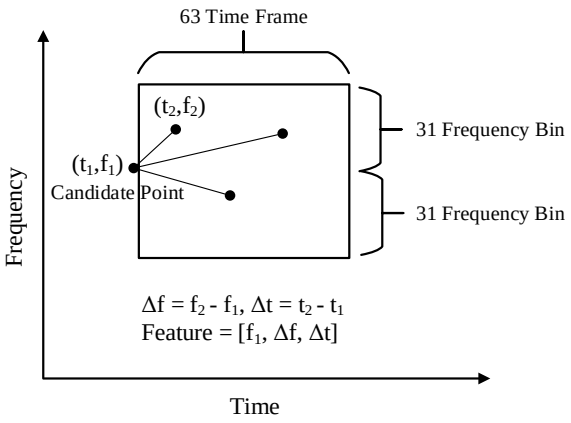

Figure 6. The combinatorial association of a pair of two points

3.1.3. Audio fingerprint

For the landmark as mentioned above, the audio fingerprint can be defined as follows:

$$Fingerprint = [f_1, \Delta f, \Delta t] \quad (5)$$

where the frequency difference $\Delta f = f_2 - f_1$, and the time difference between the two points $\Delta t = t_2 - t_1$. The fingerprint is also associated with the offset time from the beginning of the audio file to the starting time t_1.

This fingerprint feature $[f_1, \Delta f, \Delta t]$ is used to generate hash code in [3]. The hash model can be defined as shown in (6).

$$f_1 \times 2^{12} + \Delta f \times 2^6 + \Delta t \quad (6)$$

where a fingerprint hash is composed of 8-bit frequency f_1, 6-bit frequency difference Δf, and 6-bit time difference Δt. Figure 7 shows an example of 20-bit hash address calculated from (6).

$[f_1, \Delta f, \Delta t]$ Hash Function Hash Address

$[216, 18, 36] \longrightarrow f_1 \times 2^{12} + \Delta f \times 2^6 + \Delta t \longrightarrow 885924 = 11011000010010100100$

Figure 7. An example of 20-bit hash address

For a 16-bit fingerprint hash, it is composed of 6-bit frequency f_1, 5-bit frequency difference Δf, and 5-bit time difference Δt. The hash model can be defined as shown in (7).

$$f_1 \times 2^{10} + \Delta f \times 2^5 + \Delta t \quad (7)$$

After the hash code is calculated, the system then uses this code as an index for searching in the database. An exact matching algorithm is applied in [3]. Unlike the Shazam algorithm, we develop a deep neural network with multiple hierarchical layers of nonlinear and linear transformations to learn compact codes from these fingerprint features such that the similarity between samples is preserved. The details are described further in the next section.

3.2. Unsupervised similarity-preserving hashing (USH)

In this paper, the hash transformations are created by an unsupervised deep neural network. As shown in Figure 8, there are 5 layers in our deep network: the input layer consists of 20 nodes of input x_i, the three hidden layers consist of 19, 18, and 16 nodes respectively, and there are 20 nodes of $\widehat{x_i}$ in the output layer.

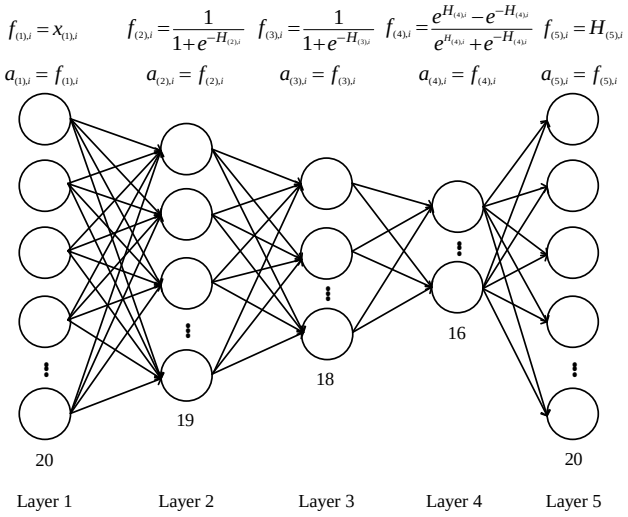

Figure 8. Our proposed unsupervised similarity-preserving hashing network (USH)

Our deep network is learned so that the output of the fourth layer can be used as the binary hash codes. For the network design, each node is composed of one input summation function and one output transformation function. The function f(·) is used to combine information by the links from other nodes, as shown in (8).

$$node_{in} = f\left(x_{(l),1}, x_{(l),2}, \dots, x_{(l),n};\ w_{(l),1}, w_{(l),2}, \dots, w_{(l),n}\right) \quad (8)$$

where $x_{(l),1}, x_{(l),2}, \dots, x_{(l),n}$ are the inputs to the node, $w_{(l),1}, w_{(l),2}, \dots, w_{(l),n}$ are the associated weights, l indicates the layer number, and n is the number of input nodes. The output activation function $a\left(f(\cdot)\right)$ is shown in (9).

$$node_{out} = a_{(l)}(node_{in}) = a_{(l)} f(\cdot) \quad (9)$$

Let H_i be the sum of product of input x_i and weight w_i, $H_i = \sum w_i x_i$, the functions of the nodes from layer 1 to layer 5 of our proposed network are defined as follows:

Layer 1: In this layer, the nodes only convey inputs to the nodes of the next consecutive layer. The functions of the ith node are shown as;

$$f_{(1),i} = x_{(1),i} \text{ and } a_{(1),i} = f_{(1),i} \quad (10)$$

Layer 2: For this layer, the sigmoid function is used as activation function. Let $x_{(2),i} = a_{(1),i}$ and $H_{(2),i} = \sum w_{(2),i}\, x_{(2),i}$. Thus, the functions of the ith node are defined as;

$$f_{(2),i} = \frac{1}{1+e^{-H_{(2),i}}} \text{ and } a_{(2),i} = f_{(2),i} \quad (11)$$

Layer 3: In this layer, the sigmoid function is applied for activation function. Let $x_{(3),i} = a_{(2),i}$ and $H_{(3),i} = \sum w_{(3),i}\, x_{(3),i}$. The functions of the ith node are defined as;

$$f_{(3),i} = \frac{1}{1+e^{-H_{(3),i}}} \text{ and } a_{(3),i} = f_{(3),i} \quad (12)$$

Layer 4: The output of each node in this layer will be used as the binary codes. During training, these codes are used to reconstruct the input data at the output layer. The hyperbolic tangent function is particularly used as activation function in this layer. Let $x_{(4),i} = a_{(3),i}$ and $H_{(4),i} = \sum w_{(4),i}\, x_{(4),i}$. The functions of the ith node are defined as;

$$f_{(4),i} = \frac{e^{H_{(4),i}} - e^{-H_{(4),i}}}{e^{H_{(4),i}} + e^{-H_{(4),i}}} \text{ and } a_{(4),i} = f_{(4),i} \quad (13)$$

Layer 5: This layer is the output or reconstruction layer. To preserve the similarity between samples, thus, the target outputs are given the same as the inputs of layer 1. Let $x_{(5),i} = a_{(4),i}$ and $H_{(5),i} = \sum w_{(5),i}\, x_{(5),i}$. The functions of the ith output node are defined as;

$$f_{(5),i} = H_{(5),i} \text{ and } a_{(5),i} = f_{(5),i} \quad (14)$$

To achieve the efficient binary codes, we include constraints in the objective function so that the codes have 4 properties: (1) belonging to {1, -1}, (2) similarity-preserving, (3) independent, and (4) balancing. In this paper, the method presented in the UH-BDNN [23] is applied to optimize the objective function which is defined as follows:

$$\begin{aligned} &min_{W,b}\, Loss = \frac{1}{2N}\left\|X - \left(W_{(L-1)}Y + b_{(L-1)} \times [1]_{1\times N}\right)\right\|^2 \\ &+ \frac{\lambda_1}{2}\sum_{l=1}^{L-1}\left\|W_{(l)}\right\|^2 + \frac{\lambda_2}{2N}\left\|H_{(L-1)} - Y\right\|^2 \\ &+ \frac{\lambda_3}{2}\left\|\frac{1}{N}H_{(L-1)}H_{(L-1)}^T - I\right\|^2 + \frac{\lambda_4}{2N}\left\|H_{(L-1)}[1]_{N\times 1}\right\|^2 \end{aligned} \quad (15)$$

$$s.t. Y \in \{1,-1\}^{K\times N} \quad (16)$$

where $X \in R^{D \times N}$ is a set of N training data with D dimension, $Y \in \{1, -1\}^{K \times N}$ is output binary code of X, K is number of bits, L is number of layers, $W_{(l)}$ is weight matrix between layer $l+1$ and layer l, b is bias vector for nodes in layer $l+1$, $H_{(l)} = f_{(l)}(w_{(l-1)}H_{(l-1)} + b_{(l-1)}[1]_{1 \times n})$ is the output values of layer l, $H_{(1)} = X$, $f_{(l)}$ is activation function of layer l, and λ_1 - λ_4 are the parameters for optimizing the objective function.

The first term of (15) makes sure that the binary code allows a good reconstruction of X. The second term is a weight regularization that encourages the network to keep the weights small in order to reduce overfitting. The third term measures the equality constraint violation. The fourth term is the independence, and the fifth term is balance of the binary codes. As shown in (16) is to ensure that each bit of the binary codes belongs to {1, -1}. After the efficient codes are produced from deep-learning network, these codes are used as search index in our content-based audio retrieval system. The song ID, t_1, f_1, Δf and Δt are stored at their hash address in the database. Table 1 shows the representation of information data.

Table 1. Representation of information data

Method	Index	Information data
USH	16 bits Hash Address	Song ID, t_1, f_1, Δf, Δt
Shazam [3]	16-bit / 20-bit Hash Address	Song ID, t_1

3.3. Sequence matching

For the query step, a sequence of query features is generated to a set of compact hash codes and used for searching in the inverse lookup database. Let Q represent a set of sequences of query features, $Q = \{q_1, q_2, \dots, q_M\}$, where q_m is a query at oreder m, $m = 1, 2, \dots, M$, and M is the total number of sequences. The learning hash function $H: q_m \rightarrow h(q_m)$, is used to map the query features to binary hash codes. We can define $Q = \{q_m\}_{m=1}^{M}$ to the corresponding binary codes as follows:

$$Y = H(Q) = \{h(q_1), h(q_2), \dots, h(q_M)\} \quad (17)$$

where Y is the hash codes of Q, $Y \in \{1, -1\}^{KxQ}$, and K is the number of bits. After learning deep network, we obtain a set of items that indexed by the hash address. Let $s_m = \{\boldsymbol{x}_{m1}, \boldsymbol{x}_{m2}, \dots, \boldsymbol{x}_{mn_m}\}$ be a set of items of q_m, $\boldsymbol{x}_{mi} \in R^{5x1}$ be the information vector that are stored in the database, and n_m is the number of items of q_m. Given $S = \{s_m\}_{m=1}^{M}$ is a set of s_m where $m = 1,2, \dots, M$, the sequence matching process is shown in Figure 9.

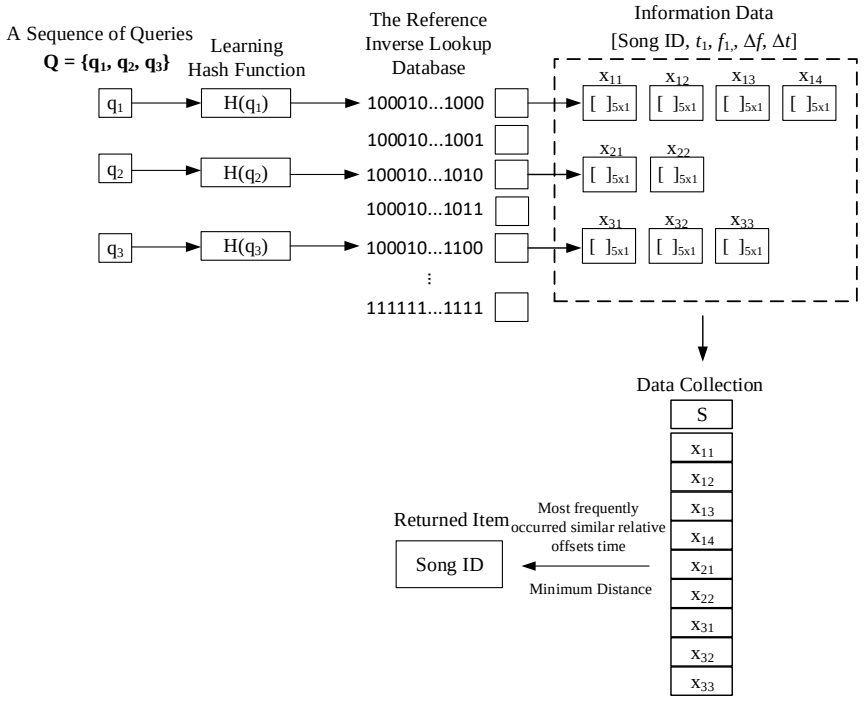

Figure 9. Our proposed sequence matching

As can be seen in Figure 9, assume that $Q = \{q_1, q_2, q_3\}$, and get the binary codes $H(Q)$, we obtain a set of s_m, $s_1 = \{x_{11}, x_{12}, x_{13}, x_{14}\}$, $s_2 = \{x_{21}, x_{22}\}$, $s_3 = \{x_{31}, x_{32}, x_{33}\}$, and $S = \{x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{31}, x_{32}, x_{33}\}$. One nearest neighbor search $NN(q_m)$ for a query item at order m from s_m is defined as follows:

$$NN(q_m) = argmin_{x_{mi} \in s_m} \|x_{mi} - q_m\|_2 \quad (18)$$

where $\|x_{mi} - q_m\|_2$ is the L_2-norm between x_{mi} and sequences q_m. Given $R_Q = \{NN(q_m)\}_{m=1}^{M}$ is a candidate items set of Q. We also apply a time offset constraint for improving the accuracy of the sequence matching. The time offset constraint $|T_{x_m} - T_{q_m}|$ is the absolute difference between T_{x_m} and T_{q_m}. It can be defined as follows:

$$|T_{x_1} - T_{q_1}| = \cdots = |T_{x_m} - T_{q_m}| \quad (19)$$

where T_{x_m} and T_{q_m} are the offset time of reference file x_m and query q_m, respectively. The constraint of offset time can be analyzed that a sequence of candidate items should occur with the same absolute difference among the time sequences. Our proposed has the following procedures.

In summary, the proposed audio retrieval algorithm is based on two parts, the similarity (minimum distance) between the audio query and the song in the reference database, and the absolute difference among the time sequences.

Algorithm: Similarity-preserving hash for content-based audio retrieval using unsupervised deep neural networks

Input:

$M = \{m_i\}_{i=1}^{N}$ N reference set;

Q query set;

Output:

SID Song ID is returned by proposed algorithm.

Step 1: Extracting fingerprint features $X = \{x_i\}_{i=1}^{N} \in R^{D \times N}$ from the reference dataset.

Step 2: Learning hash where $x_i \in R^D$ is an input vector. The objective function defined as equations 15-16. In this step, we apply the learning function $h(q)$ for audio fingerprint q in step 3.

Step 3: Sequence matching

1. The query sample is divided into M fingerprints $Q = \{q_m\}_{m=1}^{M}$
2. $s_i = \{\}, A = \{\}, S = \{\}, R_Q = \{\}$

```
for i = 1, 2, …, M do
  index_i = h(q_i)
  s_i = all items in A(index_i) are collected into s_i
  S = S ∪ s_i
end
for i = 1, 2, …, M do
  Aux = maxValue
  for j = 1, 2, …, sizeOf(s_i) do
    if ||x_j − q_i||_2 < Aux
      Aux = ||x_j − q_i||_2
      absTime = |T_{x_j} − T_{q_i}|
      SID = SID_{x_j}
    end
  end
  R_Q = R_Q ∪ {SID, absTime}
end
```

3. Finding max frequency of each Song ID of R_Q where they have same absolute difference among the time sequences.

4. EXPERIMENTAL AND PERFORMANCE ANALYSIS

4.1. Database

The performance of our proposed USH method is evaluated on the Extended Ballroom dataset freely available in [24, 25]. The dataset consists of 4,180 musical excerpts of 13 genres with a length of 30 seconds each. The audio quality of this data is 44.1kHz, 192-kbps, stereo, mp3 format. In this work, the audio signal is downsampled to 8KHz. to make the data easier to handle as previously mentioned. The training set (also used as reference database for retrieval) is composed of 3,000 tracks from 8 genres, the same rhythm class as our previous works [26, 27]. A set of 1,000 audio queries with a length of 10 seconds each are

randomly selected from those 3,000 tracks. Another set of 200 audio queries comes from audio files that do not appear in the database, in order to analyze the false positive rate. Each audio sample is represented by a 20-dimensional feature vector extracted by fingerprint algorithm. Table 2 shows the number of samples in the database and the query set.

Table 2. Audio samples in the database and query set

Set	Number of tracks	Length of segment (s)	Number of samples
Database	3,000	30	441,184
Query	1,200	10	67,785

4.2. Performance evaluation

4.2.1. Effectiveness of retrieval

On a total of 1,200 audio queries, the retrieval results obtained from our proposed USH method and state-of-the-art data-independent method, the Shazam algorithm, are shown in Table 3. The false negative (FN) refers to the incorrect identification that the query audio does not exist in the database when it does, true positive (TP) refers to the correct identification of the audio recording from the query, false positive (FP) refers to the incorrect identification of the wrong recording when the correct recording does not exist in the database, and true negative (TN) refers to the correct identification that no audio recording matches the query. According to the experimental results, we obtain higher percentage of accuracy (88.92%) for the proposed USH than state-of-the-art data-independent method, the Shazam algorithm (71.67% for 16-bit hash code, 87.42% for 20-bit hash code). Figure 10 shows the retrieval accuracy comparison between the two different methods.

Table 3. Retrieval results comparison between USH and state-of-the-art data-independent method

Method	FN	TP	FP	TN
USH 16-bit	114	886	19	181
Shazam 16-bit	290	710	50	150
Shazam 20-bit	132	868	19	181

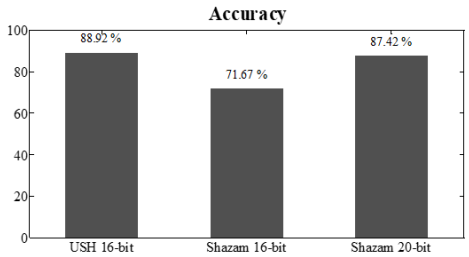

Figure 10. The retrieval accuracy of the proposed USH and the Shazam algorithm

The effectiveness of the USH is evaluated through the experiments and compared with state-of the-art data-independent method in terms of precision, recall, F1 score, and false positive rate [28], as follows:

$$Precision = \frac{TP}{TP+FP} \times 100 \quad (20)$$

$$Recall = \frac{TP}{TP+FN} \times 100 \quad (21)$$

$$\text{F1 } score = 2 \times \frac{Precision \text{ x } Recall}{Precision + Recall} \quad (22)$$

$$False\ Positive\ Rate = \frac{FP}{FP+TN} \times 100 \quad (23)$$

As can be seen in Table 4, we obtain higher precision and recall values for the proposed USH than state-of-the-art data-independent method both in 16-bit and 20-bit hash code. The F1 score (in the fourth column) shows the overall effectiveness of two different methods. Furthermore, the USH has a significantly

lower percentage of false positives (9.50%) than state-of-the-art data-independent method (25.00%) for the same 16-bit hash code. Its shows the superior performance of the USH on short codes.

Table 4. Effectiveness comparison between USH and state-of-the-art data-independent method

Method	Precision	Recall	F1 score	% False positive
USH 16-bit	97.90	88.60	93.02	9.50
Shazam 16-bit	93.42	71.00	80.68	25.00
Shazam 20-bit	97.86	86.80	92.00	9.50

4.2.2. Storage cost

As a result of our proposed USH method, the 20-bit fingerprint features can be mapped into the 16-bit binary code. Hence, it significantly reduces the database size by 16-fold, resulting in higher search performance.

4.3. Discussion

The performance of the proposed USH for large audio database retrieval is evaluated through the experiments and compared with state-of-the-art data-independent hashing method, the Shazam algorithm, on a test set of 3,000 audio recordings. The experimental results support the effectiveness of the USH with high precision and recall values at 97.90% and 88.60% respectively. For the satisfactory results, the hash codes produced from our proposed method have similarity preserving property, i.e., the similarity items are mapped to the same hash code, the dissimilarity items are mapped to another one. The data-independent methods do not take into account for this property.

The Shazam algorithm has higher percentage of false positives (25.00%) than the USH (9.50%) for the same 16-bit hash code. It shows that the Shazam algorithm is more likely to give incorrect identifications for the short-length codes and that is of inferior performance in audio retrieval. Furthermore, if the database size is increased tremendously in the future, it is most likely that the Shazam algorithm would result a significant number of false positive matches.

Let's consider the collection S of the USH and the Shazam algorithm which effect to the accuracy of audio retrieval. For the Shazam algorithm, the collection S consists of the data items where the search algorithm tries to find only the matching items for those search queries regardless of similarity preserving, and this may result in losing a number of relevant data. For the collection S of our proposed USH method consists of candidate data items where the search algorithm focuses on the similarity between the search queries and the items in the database. As shown in Table 5, with the collection S of the USH, the Song ID 48 is correctly identified by the two data items with the smallest distance (distance=1) at the same time offset. For the Shazam algorithm, the relevant song cannot be retrieved.

Table 5. Example of the collection S of the proposed USH and the Shazam algorithm

Method	Song ID	Time offset	Distance	Number of item(s)
USH 16-bit	150	128	16	2
	1041	239	28	2
	936	-232	32	2
	48*	**152**	**1**	**2**
	306	228	48	2
	2453	516	24	2
	690	3720	32	1
	690	2726	32	1
Shazam 16-bit	2732	164	-	2
	847	19	-	2
	684	2871	-	1
	674	4475	-	1
	675	2584	-	1
	676	1564	-	1
	676	4739	-	1
	677	1820	-	1
Shazam 20-bit	1174	52	-	1
	1176	557	-	1
	1230	218	-	1
	706	363	-	1
	340	395	-	1
	48	153	-	1
	93	448	-	1
	139	453	-	1

* Refers to the system correctly identifies the audio recording

The major factors for our superiority are that 1) the similarity-preserving hash codes produced from our proposed USH method, and 2) the audio retrieval algorithm proposed composes of 2 metrics, one is the L_2-norm, and the other is absolute offset time difference. These factors increase the ability to identify the candidate items according to the similarity level of audio sample and the songs in the reference database. And this significantly improves the retrieval performance.

5. CONCLUSION

In this paper, an unsupervised similarity-preserving hashing (USH) method for content-based audio retrieval is proposed. We develop a deep network with multiple hierarchical layers of nonlinear and linear transformations to learn compact hash codes where the similarity between samples is preserved. The independence and balance properties are included and optimized in the objective function to improve the codes. The experimental results on the Extended Ballroom dataset show the superiority of our proposed method over state-of-the-art data-independent method. It is suggested future work should be focused on extending USH to supervised hashing by leveraging the semantic labels to enhance the retrieval performance.

REFERENCES

[1] P. Grosche, M. Müller, and J. Serrà, "Audio Content-Based Music Retrieval," *Multimodal Music Processing. Dagstuhl Follow-Ups*, vol. 3, pp. 157-174, 2012.
[2] J. Haitsma, and T. Kalker, "A highly robust audio fingerprinting system with an efficient search strategy," *Journal of New Music Research*, vol. 32, no. 2, pp. 211-222, 2003.
[3] A. L. Wang, "An Industrial-Strength Audio Search Algorithm*," 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 7-13, 2003.
[4] P. Panyapanuwat, S. Kamonsantiroj, and L. Pipanmaekaporn, "Time-Frequency Ratio Hashing for Content-Based Audio Retrieval," *2017 9th International Conference on Knowledge and Smart Technology (KST)*, Chonburi, pp. 205-210, 2017.
[5] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," *25th International Conference on Very Large Data Bases (VLDB'99)*, pp. 518-529, 1999.
[6] B. Kulis, P. Jain, and K. Grauman, "Fast Similarity Search for Learned Metrics," In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2143-2157, 2009.
[7] M. Raginsky, and S. Lazebnik, "Locality-Sensitive Binary Codes from Shift-Invariant Kernels," *23rd Annual Conference on Neural Information Processing Systems (NIPS'09)*, pp. 1509-1517, 2009.
[8] Y. Zheng, J. Zhuand, W. Fangand, and L.-H. Chi, "Deep Learning Hash for Wireless Multimedia Image Content Security," *Journal of Security and Communication Networks*, vol. 2018, pp. 1-13, 2018.
[9] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to Hash for Indexing Big Data-A Survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34-57, 2016.
[10] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," *21st International Conference on Neural Information Processing Systems (NIPS'08)*, pp. 1753-1760, 2008.
[11] B. Kulis, and K. Grauman, "Kernelized Locality-Sensitive Hashing for Scalable Image Search," *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, Kyoto, pp. 2130-2137, 2009.
[12] Y. Gong, and S. Lazebnik, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes," *IEEE Conference on Computer Vision and Pattern recognition (CVPR 2011)*, Providence, RI, pp. 817-824, 2011.
[13] B. Kulis, and T. Darrell, "Learning to Hash with Binary Reconstructive Embeddings," *22nd International Conference on Neural Information Processing Systems (NIPS'09)*, pp. 1042-1050, 2009.
[14] W. Liu, J. Wang, R. Ji, Y. G. Jiang, and S. F. Chang, "Supervised Hashing with Kernels," *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, Providence, RI, pp. 2074-2081, 2012.
[15] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming Distance Metric Learning," *25th International Conference on Neural Information Processing Systems (NIPS'12)*, pp. 1061-1069, 2012.
[16] J. Wang, S. Kumar, and S. F. Chang, "Semi-Supervised Hashing for large-Scale Search," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393-2406, 2012.
[17] F. Shen, C. Shen, W. Liu W, and H. T. Shen, "Supervised Discrete Hashing," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, pp. 37-45, 2015.
[18] X. Bai, H. Yang, J. Zhou, P. Ren, and J. Cheng, "Data-dependent Hashing Based on p-Stable Distribution," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5033-5046, 2014.
[19] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A Survey on Learning to Hash," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769-790, 2018.
[20] J. He, S. F. Chang, R. Radhakrishnan, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time*," IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, Providence, RI, pp. 753-760, 2011.
[21] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A Review of Audio Fingerprinting," *Journal of VLSI Signal Processing*, vol. 41, pp. 271-284, 2005.
[22] Dan Ellis, "Robust Landmark-Based Audio Fingerprinting," 2015. [Online]. Available: https://labrosa.ee.columbia.edu/matlab/fingerprint/.

[23] T. T. Do, A. D. Doan, and N. M. Cheung, "Learning to Hash with Binary Deep Neural Network," *14th European Conference on Computer Vision (ECCV 2016*), pp. 219-234, 2016.
[24] U. Marchand, and G. Peeters. "Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description," *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, Vietri sul Mare, pp. 1-6, 2016.
[25] U. Marchand, and G. Peeters, "The Extended Ballroom Dataset," *17th International Society for Music Information Retrieval Conference (ISMIR 2016) Late-Breaking Sessio*n, New-York, USA, pp. 1-3, 2016.
[26] P. Panyapanuwat, S. Kamonsantiroj, and L. Pipanmaekaporn, "Unsupervised Learning Hash for Content-Based Audio Retrieval Using Deep Neural Networks," *2019 11th International Conference on Knowledge and Smart Technology (KST)*, Phuket, Thailand, pp. 99-104, 2019.
[27] P. Panyapanuwat, and S. Kamonsantiroj, "Performance Comparison of Unsupervised Deep Hashing with Data-independent Hashing for Content-Based Audio Retrieval," *2019 2nd International Conference on Electronics, Communications and Control Engineering*, pp. 16-20, 2019.
[28] C. Manning, P. Raghavan, and H. Schütze, "An Introduction to Information Retrieval," *Cambridge University Press*, 2009.

BIOGRAPHIES OF AUTHORS

Petcharat Panyapanuwat holds a bachelor's in mathematics and a master's degree in software engineering. She is currently a Ph.D. candidate at Department of Computer and Information Science, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand. Her current research interest focuses on music information retrieval.

Suwatchai Kamonsantiroj is currently a lecturer at Department of Computer and Information Science, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand. He holds a bachelor's degree in mechanical engineering and a master's degree in information technology management. He also earned his doctoral degree in computer engineering from Kasetsart University, Thailand, graduating in 2008. His current research interests include neural network, time series analysis, and artificial intelligence.

Luepol Pipanmaekaporn is currently a lecturer at Department of Computer and Information Science, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand. He holds both a bachelor's and a master's degree in computer science. He also earned his doctoral degree in computer science from Queensland University of Technology, Australia, graduating in 2013. His current research interests include information retrieval, web mining, and data mining.