Multidim Syst Sign Process (2018) 29:887–902
https://doi.org/10.1007/s11045-017-0476-x

Deep learning of chroma representation for cover song identification in compression domain

Jiunn-Tsair Fang[1] · **Yu-Ruey Chang**[2] · **Pao-Chi Chang**[2]

Received: 26 February 2016 / Revised: 25 October 2016 / Accepted: 10 February 2017 /
Published online: 21 February 2017

Abstract Methods for identifying a cover song typically involve comparing the similarity of chroma features between the query song and another song in the data set. However, considerable time is required for pairwise comparisons. In addition, to save disk space, most songs stored in the data set are in a compressed format. Therefore, to eliminate some decoding procedures, this study extracted music information directly from the modified discrete cosine transform coefficients of advanced audio coding and then mapped these coefficients to 12-dimensional chroma features. The chroma features were segmented to preserve the melodies. Each chroma feature segment was trained and learned by a sparse autoencoder, a deep learning architecture of artificial neural networks. The deep learning procedure was to transform chroma features into an intermediate representation for dimension reduction. Experimental results from a covers80 data set showed that the mean reciprocal rank increased to 0.5 and the matching time was reduced by over 94% compared with traditional approaches.

Keywords Cover song · Music retrieval · Sparse autoencoder · Descriptor · Advanced audio coding

1 Introduction

A cover song is a new version of a previously recorded track. The cover version may be different from the original song in timbre, tempo, structure, key, or arrangement. Textual labels entered into search engines cannot identify cover songs. Content-based music retrieval technology is thus applied for identification. However, finding a match between an original song and its cover version is challenging for computers. Computers require rules or procedures to

✉ Pao-Chi Chang
pcchang@ce.ncu.edu.tw

1 Department of Electronic Engineering, Ming Chuan University, No.5, Deming Rd., Taoyuan City 33348, Taiwan

2 Department of Communication Engineering, National Central University, No.300, Jhongda Rd., Taoyuan City 32001, Taiwan

distinguish music characteristics for similarity measurement. In particular, a cover version often adopts various interpretations of the original song. The matching scheme should be robust to changes in tempo, key, timbre, and musical instruments (Ellis and Poliner 2007). Tonality and harmony are generally considered major features to represent music for similarity identification. The widely used feature is the pitch-class profile, also called "chroma," which represents the intensity of 12 semitones (Fujishima 1999).

Ellis and Poliner (2007) won the MIREX cover song identification competition in 2006. They first tracked the beat to overcome variability in tempo and then collected 12-dimensional chroma features to overcome variations in timbre. Finally, they used cross-correlation to compute similarities between songs. Lee (2006) used hidden Markov models to transform chroma features into chord sequences and applied dynamic time warping (DTW) to determine the minimum alignment cost. Sailer used a pitch line to align melody and measured the melodic similarity (Sailer and Dressler 2006). A vector quantization method was applied to chroma features for similarity calculation (Riley et al. 2008).

To save disk space, most songs stored in the data set are in a compressed format. It would be a challenge in audio processing to design a single time–frequency representation for both audio coding and indexing (Ravelli et al. 2010). The concept of transform-domain audio indexing has been studied for different audio standards (Patel and Sethi 1996; Yapp and Zick 1997; Kiranyaz et al. 2006). Indexing audio signals in the compression domain can eliminate some decoding procedures to reduce the computational complexity.

Working with MP3s, Tsai and Wang (2004) subtracted the scale-factor and subband coefficients through partial decoding from the inverse modified discrete cosine transform (MDCT). Quantization-tree indexing and melody-line pitch tracking were further applied as retrieval features (Tsai and Chang 2009). Advanced audio coding (AAC), finalized in 1997, is a widely used audio coding standard (ISO/IEC 1997). It was designed to be the successor of the MP3 which was the audio standard of MPEG-2. Ravelli et al. (2010) developed an audio codec based on MDCT from the transform-domain of AAC for beat tracking, chord recognition, and musical genre classification. Our previous work partially decoded the MDCT coefficients from AAC to extract chroma features for similarity comparison between two songs. It also showed that the segmentation of chroma features could improve the matching performance. This is because the segmentation preserves the melody of a song (Chang et al. 2013).

In information retrieval, features are often selected according to experience and knowledge. Manual feature selection is typically subjective and often time consuming. To reduce the subjectivity and increase the efficiency, machine learning technology has been widely used. Some machine learning algorithms can learn by using examples as inputs and building a model that makes data-driven predictions or decisions.

Artificial neural networks (ANN) learn and acquire knowledge by connecting neural units that mimic the basic structure of systems of biological neurons. Many neural units can operate in parallel within a single ANN. ANNs model the relations between inputs and outputs nonlinearly, making ANNs more powerful than the principal component analysis (PCA) method. The importance or weight of each feature of an ANN can be learned objectively.

Deep learning refers to a class of machine learning techniques with many layers of information processing for neural units learning. In 2006, Hinton and Salakhutdinov (2006) proposed extension of ANNs with deep layers to produce deep belief networks (DBNs). DBNs are composed of multiple logistic regressions and restricted Boltzmann machines (RBMs) (Hinton et al. 2006). RBMs can be expressed using bipartite graphs with visible and hidden variables, forming two layers of vertices, with no connections between units of the same layer (Smolensky 1986). A greedy layerwise approach (Bengio et al. 2007), Gibb sampling (Casella and George 1992), and contrastive divergence (Mnih and Hinton 2005) have been applied for

layer-by-layer pretraining of DBNs, and thus RBMs can be stacked and trained to form DBNs. Finally, the method of back-propagation fine-tunes the weighting of each neural unit. DBNs have shown favorable performance in image retrieval and speech recognition (Nair and Hinton 2009; Dahl et al. 2010; Hinton 2012).

An autoencoders (AE) refers to a class of deep learning, and its neural units can be trained without supervision. The special character of an AE is that its output targets equal its input values. The number of hidden variables can be constrained to be less than the number of inputs (Al-Shareef et al. 2008). Some variants of AEs have been proposed, but the sparse AE (SAE) has the advantage of over-completed representation, which means that the hidden layer can be larger than the input layer, but only a few elements are not zero (Ranzato et al. 2007).

In this study, modified discrete cosine transform coefficients from AAC-encoded music were directly extracted and mapped into chroma features. The chroma features were segmented to preserve the melodies. To shorten the considerable time required for pairwise comparisons, SAEs were employed to reduce the dimensions of the chroma features in each segment.

The remainder of this paper is organized as follows. Section 2 describes related studies on cover song identification and SAEs. The deep learning methods for the proposed chroma feature transformation are described in Sect. 3. Experimental results are described in Sect. 4. Finally, Sect. 5 provides a conclusion.

2 Related work

The conventional methods for cover song identification, AAC, and deep learning are described in this section. The cover song identification method includes chroma feature patching, the optimal transpose index (OTI), a similarity matrix, and dynamic alignment method. The description of AAC focuses on the physical meaning of MDCT coefficients, and the discussion of deep learning focuses on SAE.

2.1 Traditional method for cover song identification

Cover songs typically preserve some aspects of the chords and harmony of the original song. The widely used feature in music retrieval is the pitch-class profile, or chroma. Chromas profile 12 semitones (C, C#, D, D#, E, F, F#, G, G#, A, A#, and B) that describe a 12-bin octave-independent histogram of energy intensity (Shepard 1982).

Cover songs typically change the key of the original composition to adapt to a different singer or instrument. Thus, transposing two songs into the same key is necessary for chroma similarity measurement. OTI is a commonly used method. The OTI method entails summing the individual intensity of each semitone from the first frame to the end frame. There are 12 semitones. Each summed semitone is then divided by the maximum value among these 12 semitones to normalize the semitone. These final 12 semitones, called global chroma features, are used to adjust the key by using the OTI function, which is defined as (1) (Ellis 2006),

$$K = \mathrm{OTI}(\mathbf{h}^d, \mathbf{h}^q) = \arg\max_J \{\mathbf{h}^d \cdot circshift(\mathbf{h}^q, J)\} \tag{1}$$

where $\mathbf{h}^q$ and $\mathbf{h}^d$ correspond to the global chroma features for the query song and the reference song, respectively, $circshift(\mathbf{h}^q, J)$ is a function that rotates the vector $\mathbf{h}^q$ with J positions, and J is an integer from 0 to 11. After estimation of the key difference distance according to

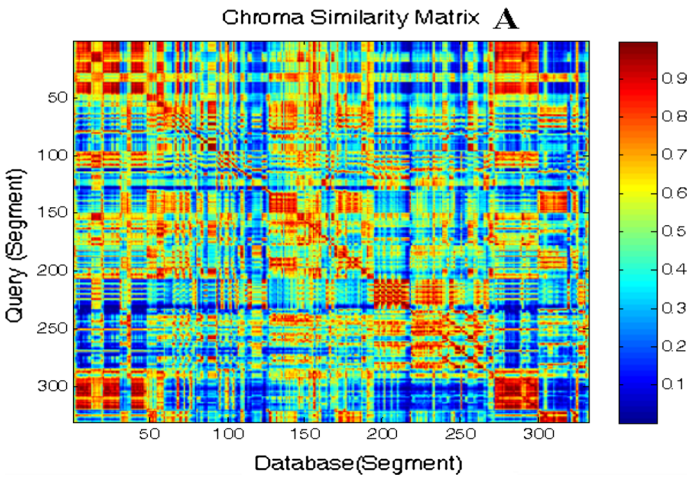

Fig. 1 Similarity matrix for comparison of two songs

the OTI function, the chroma features of the query song can be transposed to the key of the reference song.

A chroma similarity matrix is usually applied to score the similarity between two songs.

Global chroma features are normalized so that all entries in the chroma similarity matrix are between 0 and 1. The closer the entry is to 1, the more similar the two clips are. Figure 1 shows the similarity matrix for an example of a query song and a reference song in a database. The result shows that these two songs are similar.

To solve the problem of tempo variation, a dynamic alignment method was applied in the chroma similarity matrix (Serra et al. 2008). This method was proposed by Waterman and Smith (1978), and was similar to the DTW method. A matrix G with dimension $(M+1) \times (E+1)$ is created, where M and E are the numbers of clips in the query song and reference song. Entries in matrix G can be calculated through a recursive formula as expressed by (2) and (3) (Waterman and Smith 1978). The recursive procedure can identify the tempo difference between two songs.

$$G(i,j) = \begin{cases} D \ \ for\ i : 2, \ldots, M+1, j : 2, \ldots, E+1 \\ 0 \ \ otherwise \end{cases} \tag{2}$$

$$D = \max \begin{cases} G(i-1, j-1) + A(i-1, j-1) \\ G(i-1, j-2) + A(i-1, j-1) \\ G(i-2, j-1) + A(i-1, j-1) \end{cases} \tag{3}$$

2.2 Advanced audio coding

The description of AAC focuses on the physical meaning of MDCT coefficients. Figure 2 shows a partial decoding process for AAC. After decoding, MDCT coefficients are processed using temporal noise shaping (TNS) and perceptual noise substitution (PNS). TNS uses a prediction approach in the frequency domain that aims at shaping the quantization noise in the time domain. PNS models the noise like components by using a parametric approach (Ravelli et al. 2010). The AAC decoder first recovers the MDCT coefficients through Huffman decoding, inverse quantization, and scaling. The coefficients are then processed using TNS

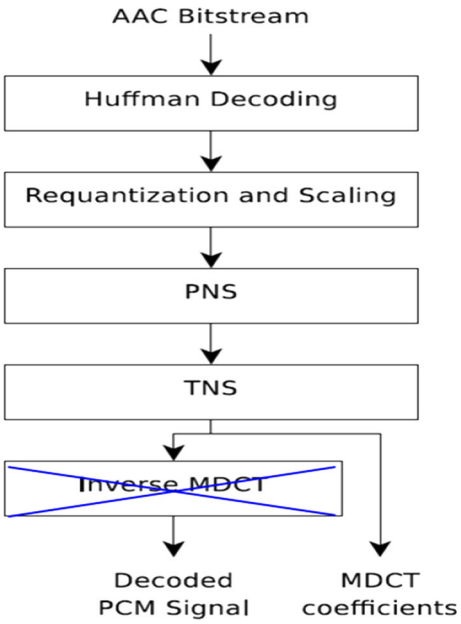

Fig. 2 Flowchart of partial decoding for AAC

and PNS. Finally, without use of an inverse time-varying MDCT, MDCT coefficients are directly extracted.

In the AAC encoding process, the time–frequency analysis tool is MDCT, which is defined as follows (ISO/IEC 1997):

$$X_{MDCT}(k) = 2\sum_{n=0}^{N-2} z(n)\cos\left(\frac{2\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{4}\right)\right), \quad 0 \le k \le \frac{N}{2},$$
$$f_k = \frac{k}{N/2}\times\frac{f_s}{2} \tag{4}$$

where x is the windowed input sequence, n is the sample index, k is the spectral coefficient index, and N is the length of the transform window. Although (4) is not equivalent to a fast Fourier transform, each MDCT coefficient $X(k)$ represents the energy magnitude of a frequency band, and therefore the MDCT coefficients can be simply allotted to chroma bin b as defined by (5):

$$Bin(b) = \mathrm{mod}\left(round\left(12\log_2\left(\frac{f}{f_0}\right)\right), 12\right) + 1 \tag{5}$$

where f is the corresponding frequency of $X(k)$, and f_0 is 16.352 Hz, represented by C0, the lowest pitch frequency. Software records the chroma feature, which is the energy intensity associated with each of the 12 semitones.

2.3 Autoencoder

An RBM model can be represented as the composition of a function that maps input data to a hidden layer and a function that maps the hidden layer to an output representation. The

question is how to set the latent variables in the hidden layer to classify each input as the correct output.

Similar to the RBM model, the hidden layer of an AE encoder can be expressed as follows:

$$h_i = sigmoid\left(\sum_j w_{i,j}x_j + d_i\right). \tag{6}$$

The reconstructed output variable can be expressed as follows:

$$\begin{aligned} x'_j &= sigmoid\left(\sum_j w'_{i,j}h_i + b_i\right) \\ &= sigmoid\left(\sum_j w'_{i,j}\left(sigmoid\left(\sum_j w_{i,j}x_j + d_i\right)\right) + b_i\right) \end{aligned} \tag{7}$$

where w and w' can be the same. The latent variables w and w' can be determined by minimizing the error function, which is the square of the difference between w and w', and b_i and d_j are the weights for the bias terms of the visible and hidden layers, respectively.

The dimensions of hidden layers are smaller than those of input layers. In other words, fewer variables are used to represent input data. Equation (7) is similar to that of two-layered networks, and thus the latent variables can be trained using the back-propagation algorithm, with the same input and output, to modify the weights of hidden layers.

The major difference between DBNs and AEs is that the output representation in AEs is the same as that of the input signals. The hidden layers can compress the input data, and the mapping from a hidden layer to an output layer can decode the original input data. Another difference between DBNs and AEs is that hidden variables in an AE must be continuous values between 0 and 1, whereas hidden variables in DBNs are either 0 or 1, depending on their probability. Because AEs use a deterministic model to obtain hidden layers, they cannot form a deep generative model, but AE parameters are easier to train. Furthermore, AEs use a nonlinear method and a sigmoid function, as shown in (6) and (7). By contrast, PCA uses a linear method to calculate eigenvalues to compress the input signals (Bengio et al. 2013; Bengio 2009).

A sparsity term can be added to an AE to create an SAE. The sparsity term can be written as follows (Andrew 2011):

$$J_{sparse}(K) = J(k) + \beta\sum_{j=1}^{s_2} KL(\rho||\hat{\rho}) \tag{8}$$

where $J_{sparse}(K) = \frac{1}{2}||e_q(k)||^2$ is the cost function, β determines the sparsity, $KL(\rho||\hat{\rho})$ is the Kullback–Leibler (KL) divergence, s_2 is the number of neurons in the hidden layer, and the index j represents a sum over the hidden units. The KL divergence can be written as follows:

$$KL(\rho||\hat{\rho}) = \rho\log\frac{\rho}{\hat{\rho}} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}} \tag{9}$$

where $\hat{\rho}$ is the average weight of the neuron, and ρ is the expected value. Sparsity is achieved by letting the average weight, $\hat{\rho}$, be constrained to the expected weight, ρ, as in (8), which constrains sparsity.

Deep learning technology is very powerful machine learning. However, large network structures cause overfitting problems for deep learning systems. The dropout method, which

entails randomly dropping units along with their connections, is to prevent complex co-adaptations to the training data (Hinton et al. 2012; Srivastava et al. 2014).

3 Proposed method for cover song dentification

In addition to the traditional method for cover song identification, the added procedures of the proposed method can be classified into three steps: the preprocessing procedure, the training procedure, and the test procedure. A block diagram of the proposed method is shown in Fig. 3.

The preprocessing procedure includes partial decoding, chroma feature extraction, and segmentation, which are described in Sect. 2.2. To avoid mapping a harmonic into the incorrect bin, the frequency bandwidth was truncated between 130 and 1000 Hz (Muller et al. 2011).

Audio signals are continuous over time, and frames, units for DFT, are highly correlated. An average 3.5-min song, sampled at a sampling rate of 16 kHz, has approximately 3000 frames. To preserve melodies for similarity comparison, we can merge several frames into a segment and some frames can overlap between two consecutive segments (Chang et al. 2013). The segment length and the overlap ratio were tested in the experiment.

As shown in Fig. 3, each song in the training data set was preprocessed; the chroma features were extracted from the MDCT coefficients of AAC. The chroma bin energy was then normalized to enhance the energy difference. The bin energy normalization is written as (10),

$$E^*(b_i, m) = \frac{E(b_i, m)}{\sum_{i=1}^{12} E(b_i, m)} \tag{10}$$

where $E(b_i, m)$ is chroma bin energy in the i th bin at the m th segment. An example of chroma bin normalization is illustrated in Fig. 4.

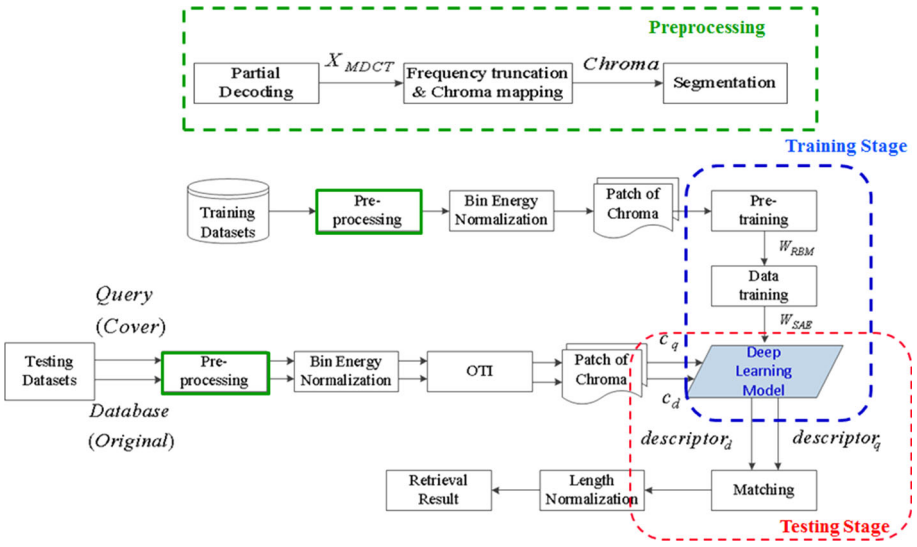

Fig. 3 Block diagram of the proposed method (Color figure online)

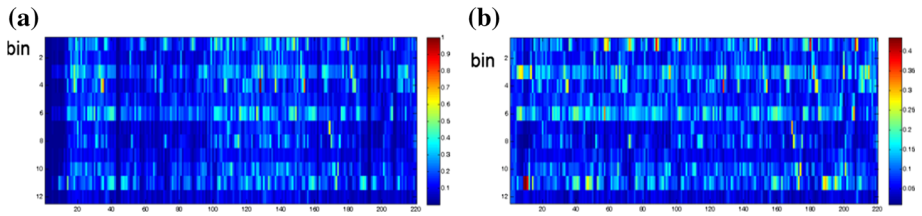

Fig. 4 Bin energy normalization: **a** before normalization, and **b** after normalization

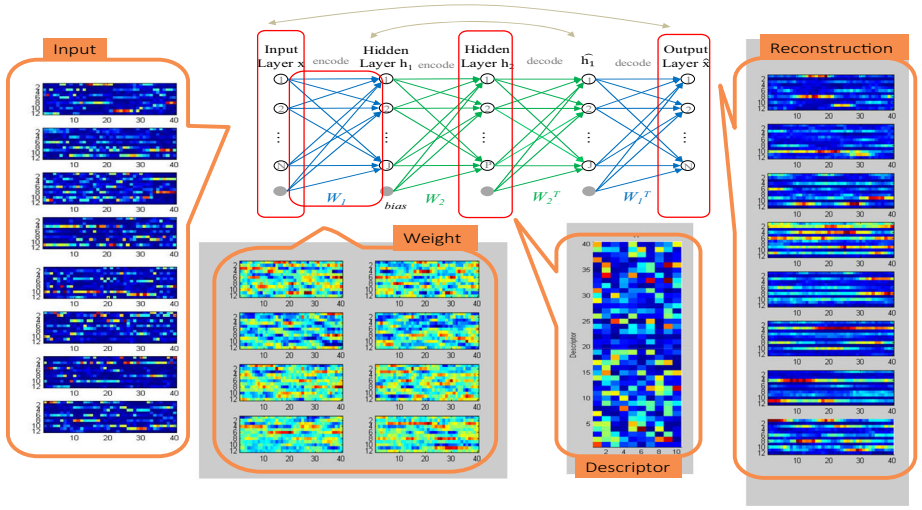

Fig. 5 An example illustrating the transformation of chroma features by an SAE

In Fig. 3, the training procedure, marked by the blue dashed lines, is to determine the parameters in the SAE. A three-layered SAE was applied in this study to reduce the dimensions of chroma features. The dimension of the first hidden layer of the SAE was the dimension of the chroma features, and the dimension of the final hidden layer of the SAE was the dimension of the output descriptor. The dimension and weights of each of the hidden layers were determined through the experiment. Figure 5 illustrates how the SAE transforms chroma features into intermediate representation of descriptors. The input data for each patch have 12 bins and 40 segments for each bin; thus, each input has a dimensionality of 480. This dimension is equal to the dimension of the first hidden layer, as shown in the bottom left portion of Fig. 5. Weights in the final hidden layer are descriptors whose dimensions are 1×40. In other words, the compression rate is 1/12. The reconstruction data are plotted on the right side, and each patch has a dimensionality of 12×40, which is the same as that of the input data. In this example, the compression rate reaches 1/12. To achieve this result, the SAEs were configured with sparsity set to 0.1 in our deep learning configuration. That is, 10% nodes in the network have non-zero connections.

The final step is the testing procedure, marked by the red dashed lines as shown in Fig. 3. After preprocessing and bin normalization, OTI was applied for normalizing the tempo variation. The test procedure included chroma feature reduction and similarity matching. Chroma features were patched, and the SAE then used a learning process to reduce the dimensions to match the descriptors. Descriptors in the query song were compared with

descriptors of songs in the data set. The matching procedure for the cover song required pairwise comparison with each song in the data set: each cover song was compared with all of the descriptors in a song, and this matching procedure was iteratively repeated until all songs in the data set had been compared.

The dynamic alignment method (Waterman and Smith 1978) was applied for similarity scoring, which accumulated the score of all of the descriptors in a song. However, each song had a distinct duration. Songs of different lengths may have different numbers of patches when subjected to the dynamic alignment method, and that might distort similarity scores. Hence, a postprocessing procedure was added to regulate the length of each song. In other words, the raw similarity score was divided by the length of the reference song, which yielded the refined score given by (11):

$$DA(Q, D)* = \frac{DA(Q, D)}{length(D)} \tag{11}$$

where DA is the original score by the dynamic alignment method, Q represents the query song, and D represents the reference song.

In addition, a method called mean reciprocal rank (MRR) was used to evaluate the performance of the music retrieval software (Voorhees 1999). For MRR, the reference songs in the database are classified into different ranks, or top N lists. Songs with different ranks are then assigned different scores. The score is the reciprocal value of its rank. For example, if a reference song is recorded in rank n, the correct retrieval for this song is given a $1/n$ score. The MRR score is obtained using (12):

$$MRR = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{rank(Query_q)} \tag{12}$$

where Q is the retrieval number and $rank(query_q)$ is the rank list of the qth retrieved song.

4 Experimental results

Experiments were conducted to evaluate the performance of the proposed method. System parameters were determined, and the system performance levels were compared with those of other methods. The database we used calls covers80 which contains 80 pop songs and 80 cover songs (http://labrosa.ee.columbia.edu/projects/coversongs/covers80/).

The experimental environment is described as follows. We used a personal computer with an Intel Core i7-3770 CPU operating at 3.4 Hz. The software was Matlab2013a, and the deep learning tools (http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox) were applied. Table 1 lists parameters for deep learning. Param-

Table 1 Parameters for deep learning

Hidden layer dimensionality	480-480-20
Learning rate	1
Epoch	100
Momentum	0.4
Sparsity	0.1
Dropout	0.5

Table 2 Preliminary performance of MRR and matching time

Training (segment)	Descriptor	Testing data (segment)									
		8		12		16		20		24	
		MRR	Time	MRR	Time	MRR	Time	MRR	Time	MRR	Time
8	8	0.396	19.617	0.416	8.400	0.406	4.621	0.396	2.908	0.417	1.982
12	12	0.441	18.714	0.480	8.138	0.463	4.423	0.436	2.795	0.437	1.905
16	16	*0.483*	*19.377*	0.466	8.264	0.422	4.568	0.441	2.866	0.418	1.950
20	20	0.450	19.336	0.442	8.308	0.438	4.583	0.439	2.869	0.451	1.958
24	24	0.442	19.535	0.462	8.304	0.465	4.555	0.428	2.857	0.437	1.990

Italic values indicate the highest MRR value

Table 3 Performance levels of proposed method

Training (segment)	Descriptor	Testing data (segment)									
		8		12		16		20		24	
		MRR	Time	MRR	Time	MRR	Time	MRR	Time	MRR	Time
8	8	0.361	82.104	0.383	34.289	0.375	19.146	0.381	12.250	0.369	8.270
12	12	0.399	80.406	0.441	37.284	0.442	20.596	0.444	12.820	0.417	8.800
16	16	0.432	78.541	0.462	34.019	0.481	18.662	0.484	11.627	0.459	8.152
20	20	0.441	78.553	0.462	34.425	0.491	18.775	*0.499*	*11.678*	0.478	8.007
24	24	0.420	78.839	0.451	34.499	0.474	18.821	0.487	11.712	0.475	7.971

Italic values indicate the highest MRR value

Table 4 Performance of PCA

PCA descriptor Training (segment)	Descriptor	Testing data (segment)									
		8		12		16		20		24	
		MRR	Time	MRR	Time	MRR	Time	MRR	Time	MRR	Time
8	8	0.348	19.511	0.405	8.408	0.430	4.625	0.414	2.941	0.417	2.007
12	12	0.391	19.741	0.407	8.436	0.422	4.667	*0.432*	*2.993*	0.416	2.038
16	16	0.391	19.468	0.413	8.346	0.429	4.617	0.427	2.903	0.422	1.999
20	20	0.391	19.750	0.412	8.398	0.410	4.636	0.402	2.929	0.415	2.011
24	24	0.390	19.497	0.413	8.384	0.413	4.626	0.426	2.911	0.420	1.999

Italic values indicate the highest MRR value

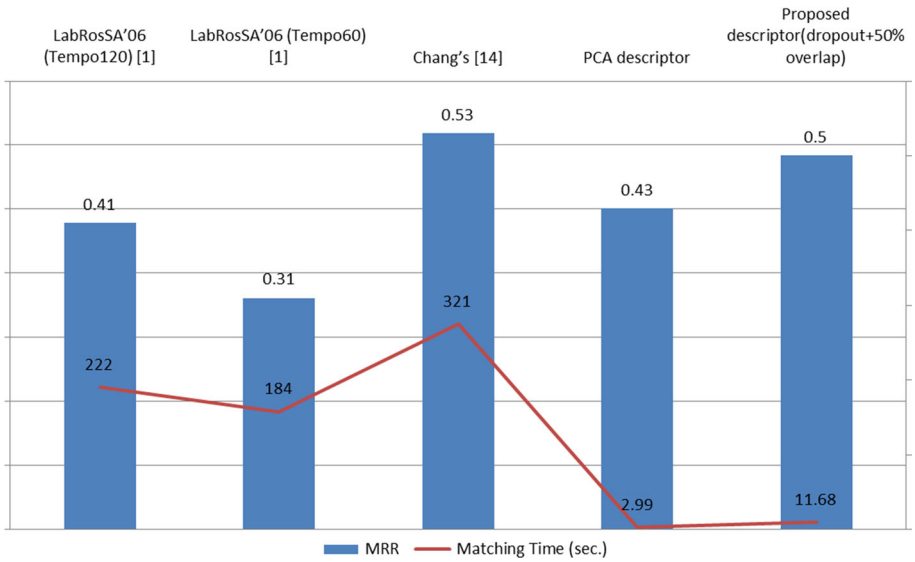

Fig. 6 Performance comparison with other systems

eters for deep learning can refer to http://www.cs.toronto.edu/~rsalakhu/talks/NLDR_NIPS06workshop.pdf and Salakhutdinov (2009).

Experiments were conducted to determine system parameters and performance. Because no training songs were available in covers80, 700 pop songs that were not part of covers80 were used in this study as training data. Table 2 lists the preliminary performance of MRR and matching time.

To improve the performance, methods of segmentation overlap was considered. After testing three kinds of overlaps, 25, 50, and 75, 50% segment of overlap was selected. Table 3 lists the MRR and matching time values. The optimum descriptor length was 20, and the optimum frame segment number for both the training and test data was 20. Weights in the final hidden layer were descriptors whose dimensions were 1×20. The highest MRR was 0.50, and its corresponding matching time was 11.2 s.

For comparison purpose, the PCA method was simulated, and its highest MRR performance was selected based on the same procedure as the proposed method. The performance of PCA method is listed in Table 4. The proposed method yielded higher MRR in most cases as well as longer matching time.

The performance of the proposed method was compared with that of other systems, namely LabROSA'06 (Tempos 120 and 60), the PCA method, and the method proposed in Chang et al. (2013). All systems were tested using the covers80 data set. In our previous work (Chang et al. 2013), our method extracted chroma features from AAC-encoded music, but chroma features did not benefit from the application of deep learning. Figure 6 shows the performance levels of MRR and similarity matching time. The proposed method improved the MRR performance, but used approximately 5% of the matching time required by LabROSA'06 (Tempo 120). The MRR performance of the method in Chang et al. (2013) was the highest among the methods, but it required an excessive length of time. PCA required the least time, but its performance was lower than that of the proposed method. PCA's lackluster performance can be ascribed to

the linear transformation it uses to reduce the chroma feature dimensionality; deep learning performs higher than linear transformation for fast chroma feature retrieval.

Compared with traditional methods for cover song identification, the method of chroma features segmentation can improve the mating performance, but it increases the matching time. With the aid of deep learning, the matching time can be saved with little decrement of matching performance.

5 Conclusion

In this work, the intermediate representation with lower dimensional features was applied to reduce the computation of pairwise comparisons between songs and cover songs. Chroma features were first extracted from the AAC compression domain to eliminate some decoding computations. Chroma features were segmented to preserve melodies. Finally, each chroma feature segment was used to train and learn from a deep learning procedure for intermediate representation with lower dimensions. Dimensionality was reduced using a sparse autoencoder. Experimental results showed that the proposed method increased the MRR to 0.50 and reduced matching time by more than 94% compared with traditional approaches. Chroma feature learning from chroma representations performs favorably for cover song identification.

References

Al-Shareef, A. J., Mohamed, E. A., & Al-Judaibi, E. (2008). One hour ahead load forecasting using artificial neural network for the western area of Saudi Arabia. *International Journal of Electrical and Computer Engineering*, *3*(13), 834–840.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Proceedings of the Advances in Neural Information Processing Systems* (pp. 153–160).

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends and in Machine Learning*, *2*(1), 1–127.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Casella, G., & George, E. I. (1992). Explaining the Gibbs sampler. *The American Statistician*, *46*(3), 167–174.

Chang, T. M., Chen, E. T., Hsieh, C. B., & Chang, P. C. (2013). Cover song identification with direct chroma feature extraction from AAC files. In *Proceedings of GCCE*, Tokyo (pp. 55–56).

Dahl, G. E., et al. (2010). Phone recognitionwith the mean-covariance restricted Boltzmann machine. *Advances in Neural Information Processing Systems*, *23*, 469–477.

Ellis, D. (2006). Beat tracking with dynamic programming. In *MIREX 2006 audio beat tracking contest system description*.

Ellis, D. P. W., & Poliner, G. E. (2007). Identifying cover songs with chroma features and dynamic programming beat tracking. In *Proceedings of the international conference on acoustics, speech and signal processing (ICASSP)*, Honolulu, HI (pp. 1429–1432).

Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of international computer music conference*, Beijing (pp. 464–467).

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. ArXiv e-prints 1207, 580.

Hinton, G. E., et al. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Hinton, E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. S. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

ISO/IEC 13818-7. (1997). Information technology—Generic coding of moving pictures and associated audio information—Part 7: Advanced audio coding (AAC).
Kiranyaz, S., Qureshi, A. F., & Gabbouj, M. (2006). A generic audio classification and segmentation approach for multimedia indexing and retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(3), 1062–1081.
Lee, K. (2006). Identifying cover songs from audio using harmonic representation. *Music Information Retrieval Evaluation eXchange (MIREX) extended abstract*.
Matlab Central, Deep Learning Toolbox [Online]. http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox.
Mnih, A., & Hinton, G. E. (2005). Learning nonlinear constraints with contrastive backpropagation. In *2005 IEEE international joint conference on neural networks, IJCNN'05. Proceedings* (pp. 1302–1307).
Muller, M., Ellis, D. P. W., Klapuri, A., & Richard, G. (2011). Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, *5*(6), 1088–1110.
Nair, V., & Hinton, G. E. (2009). 3D object recognition with deep belief nets. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS '09 (pp. 1339–1347).
Ng, A. (2011). Sparse autoencoder. In *CS294A lecture notes*.
Patel, N., & Sethi, I. (1996). Audio characterization for video indexing. In *Proceedings of SPIE* (pp. 373–384).
Ranzato, M., Boureau, Y., & LeCun,Y. (2007). Sparse feature learning for deep belief networks. In *Advances in neural information processing systems 20 (NIPS)*.
Ravelli, E., Richard, G., & Daudet, L. (2010). Audio signal representations for indexing in the transform domain. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 434–446.
Riley, M., Heinen, E., & Ghosh, J. (2008). A text retrieval approach to content-based audio retrieval. In *Proceedings of international conference on music information retrieval*, Philadelphia, Pennsylvaia (pp. 295–300).
Sailer, C., & Dressler, K. (2006). Finding cover songs by melodic similarity. *Music Information Retrieval Evaluation eXchange (MIREX) extended abstract*
Salakhutdinov, R. (2009). Learning deep generative models. Doctoral dissertation, University of Toronto.
Salakhutdinov, R. Nonlinear dimensionality reduction using neural networks. http://www.cs.toronto.edu/~rsalakhu/talks/NLDR_NIPS06workshop.pdf.
Serra, J., G'omez, E., & Herrera, P. (2008). Transposing chroma representations to a common key. In *Proceedings of IEEE CS conference on the use of symbols to represent music and multimedia objects*, Citeseer (pp. 45–48).
Shepard, R. N. (1982). Structural representations of musical pitch. In D. Deutsch (Ed.), *The psychology of music* (1st ed.). Amsterdam: Swets & Zeitlinger.
Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland & C. PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, pp. 194–281). Cambridge, MA: MIT Press.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskeve, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.
The *Covers80* cover song data set [Online]. http://labrosa.ee.columbia.edu/projects/coversongs/covers80/.
Tsai, T. H., & Chang, W. C. (2009). Two-stage method for specific audio retrieval based on MP3 compression domain. In *Proceedings of IEEE international symposium on circuits and systems* (pp. 713–716).
Tsai, T. H., & Wang, Y. T. (2004). Content-based retrieval of audio example on MP3 compression domain. In *Proceedings of IEEE 6th workshop on multimedia signal processing* (pp. 123–126).
Voorhees, E. M. (1999). The TREC-8 question answering track report. In *Proceedings of the 8th text retrieval conference (TREC-8)*.
Waterman, M. S., & Smith, T. F. (1978). RNA secondary structure: A complete mathematical analysis. *Mathematical Biosciences*, *42*(3–4), 257–266.
Yapp, L., & Zick, G. (1997). Speech recognition on MPEG/audio encoded files. In *Proceedings of IEEE international conference multimedia computing and systems* (pp. 624–625).

Jiunn-Tsair Fang received the Ph.D. degree in electrical engineering from National Chung-Cheng University, Taiwan in 2004. Currently, he is an assistant professor in the Department of Electronic Engineering at Ming Chuan University, Taiwan. His researching interest includes video/image coding, audio processing, and channel coding.

Yu-Ruey Chang received the B.S. and M.S. degrees in communication engineering from CCU and NCU, Taiwan, in 2013 and 2015, respectively. His research interests include audio signal processing and deep learning.

Pao-Chi Chang received the B.S. and M.S. degrees from NCTU, Taiwan, and the Ph.D. degree from Stanford University, California, 1986, all in electrical engineering. From 1986 to 1993, he was a research staff at IBM T.J. Watson Research Center, New York. In 1993, he joined the faculty of NCU, Taiwan, where he is presently a professor in the Department of Communication Engineering. His main research interests include speech/audio coding, video/image compression, and multimedia retrieval.