UNSUPERVISED LEARNING OF SPARSE FEATURES FOR SCALABLE AUDIO CLASSIFICATION

Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu and Yann LeCun
Courant Institute of Mathematical Sciences
New York University
mbh305@nyu.edu ; yann@cs.nyu.edu

ABSTRACT

In this work we present a system to automatically learn features from audio in an unsupervised manner. Our method first learns an overcomplete dictionary which can be used to sparsely decompose log-scaled spectrograms. It then trains an efficient encoder which quickly maps new inputs to approximations of their sparse representations using the learned dictionary. This avoids expensive iterative procedures usually required to infer sparse codes. We then use these sparse codes as inputs for a linear Support Vector Machine (SVM). Our system achieves 83.4% accuracy in predicting genres on the GTZAN dataset, which is competitive with current state-of-the-art approaches. Furthermore, the use of a simple linear classifier combined with a fast feature extraction system allows our approach to scale well to large datasets.

1. INTRODUCTION

Over the past several years much research has been devoted to designing feature extraction systems to address the many challenging problems in music information retrieval (MIR). Considerable progress has been made using task-dependent features that rely on hand-crafted signal processing techniques (see [13] and [26] for reviews). An alternative approach is to use features that are instead learned automatically. This has the advantage of generalizing well to new tasks, particularly if the features are learned in an unsupervised manner.

Several systems to automatically learn useful features from data have been proposed over the years. Recently, Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs) and sparse coding (SC) algorithms have enjoyed a good deal of attention in the computer vision community. These have led to solid and state-of-the-art results on several object recognition benchmarks [8, 15, 23, 30].

Some of these methods have also began receiving interest as means to automatically learn features from audio data. The authors of [22] explored the use of sparse coding using learned dictionaries in the time domain, for the purposes of genre recognition. Convolutional DBNs were used in [16] to learn features from speech and music spectrograms in an unsupervised manner. Using a similar method, but with supervised fine-tuning, the authors in [12] were able to achieve 84.3% accuracy on the Tzanetakis genre dataset, which is one of the best reported results to date.

Despite their theoretical appeal, systems to automatically learn features also bring a specific set of challenges. One drawback of DBNs noted by the authors of [12] were their long training times, as well as the large number of hyperparameters to tune. Furthermore, several authors using sparse coding algorithms have found that once the dictionary is learned, inferring sparse representations of new inputs can be slow, as it usually relies on some kind of iterative procedure [14, 22, 30]. This in turn can limit the real-time applications or scalability of the system.

In this paper, we investigate a sparse coding method called Predictive Sparse Decomposition (PSD) [11, 14, 15] that attempts to automatically learn useful features from audio data, while addressing some of these drawbacks. Like many sparse coding algorithms, it involves learning a dictionary from a corpus of unlabeled data, such that new inputs can be represented as sparse linear combinations of the dictionary's elements. It differs in that it also trains an encoder that efficiently maps new inputs to approximations of their optimal sparse representations using the learned dictionary. As a result, once the dictionary is learned inferring the sparse representations of new inputs is very efficient, making the system scalable and suitable for real-time applications.

2. THE ALGORITHM

2.1 Sparse Coding Algorithms

The main idea behind sparse coding is to express signals $\mathbf{x} \in \mathbb{R}^n$ as sparse linear combinations of basis functions chosen out of an overcomplete set. Letting $\mathbf{B} \in \mathbb{R}^{n \times m}$ $(n < m)$ denote the matrix consisting of basis functions $\mathbf{b_j} \in \mathbb{R}^n$ as columns with weights $\mathbf{z} = (z_1, ..., z_m)$, this relationship can be written as:

$$\mathbf{x} = \sum_{j}^{m} z_j \mathbf{b_j} = \mathbf{Bz} \quad (1)$$

where most of the z_i's are zero. Overcomplete sparse representations tend to be good features for classification systems, as they provide a succinct representation of the signal, are robust to noise, and are more likely to be linearly separable due to their high dimensionality.

Directly inferring the optimal sparse representation $\mathbf{z}$ of a signal $\mathbf{x}$ given a dictionary $\mathbf{B}$ requires a combinatorial search, intractable in high dimensional spaces. Therefore, various alternatives have been proposed. Matching Pursuit methods [21] offer a greedy approximation to the solution. Another popular approach, called Basis Pursuit [7], involves minimizing the loss function:

$$L_d(\mathbf{x}, \mathbf{z}, \mathbf{B}) = \frac{1}{2}||\mathbf{x} - \mathbf{Bz}||_2^2 + \lambda||\mathbf{z}||_1 \quad (2)$$

with respect to $\mathbf{z}$. Here λ is a hyper-parameter setting the tradeoff between accurate approximation of the signal and sparsity of the solution. It has been shown that the solution to (2) is the same as the optimal solution, provided it is sparse enough [10]. A number of works have focused on efficiently solving this problem [1, 7, 17, 20], however they still rely on a computationally expensive iterative procedure which limits the system's scalability and real-time applications.

2.2 Learning Dictionaries

In classical sparse coding, the dictionary is composed of known functions such as sinusoids, gammatones, wavelets or Gabors. One can also *learn* dictionaries that are adaptive to the type of data at hand. This is done by first initializing the basis functions to random unit vectors, and then iterating the following procedure:

1. Get a sample signal $\mathbf{x}$ from the training set

2. Calculate its optimal sparse code $\mathbf{z}^*$ by minimizing (2) with respect to $\mathbf{z}$. Simple optimization methods such as gradient descent can be used, or more sophisticated approaches such as [1, 7, 20].

3. Keeping $\mathbf{z}^*$ fixed, update $\mathbf{B}$ with one step of stochastic gradient descent: $\mathbf{B} \leftarrow \mathbf{B} - \nu \frac{\partial L_d}{\partial B}$, where L_d is the loss function in (2). The columns of $\mathbf{B}$ are then rescaled to unit norm, to avoid trivial minimizations of the loss function where the code coefficients go to zero while the bases are scaled up.

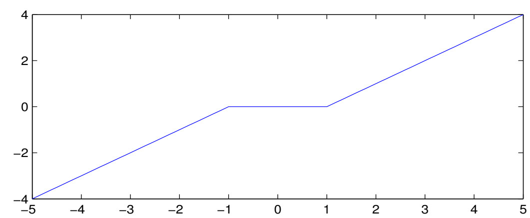

Figure 1. Shrinkage function with $\theta = 1$

There is evidence that sparse coding could be a strategy employed by the brain in the early stages of visual and auditory processing. The authors in [24] found that basis functions learned on natural images using the above procedure resembled the receptive fields in the visual cortex. In an analogous experiment [28], basis functions learned on natural sounds were found to be highly similar to gammatone functions, which have been used to model the action of the basilar membrane in the inner ear.

2.3 Predictive Sparse Decomposition

In order to avoid the iterative procedure typically required to infer sparse codes, several works have focused on developing nonlinear, trainable encoders which can quickly map inputs to approximations of their optimal sparse codes [11, 14, 15]. The encoder's architecture is denoted $\mathbf{z} = f_e(\mathbf{x}, \mathbf{U})$, where $\mathbf{x}$ is an input signal, $\mathbf{z}$ is an approximation of its sparse code, and $\mathbf{U}$ collectively designates all the trainable parameters of the encoder. Training the encoder is performed by minimizing the encoder loss $L_e(\mathbf{x}, \mathbf{U})$, defined as the squared error between the predicted code $\mathbf{z}$ and the optimal sparse code $\mathbf{z}^*$ obtained by minimizing (2), for every input signal $\mathbf{x}$ in the training set:

$$L_e(\mathbf{x}, \mathbf{U}) = \frac{1}{2}||\mathbf{z}^* - f_e(\mathbf{x}, \mathbf{U})||^2 \quad (3)$$

Specifically, the encoder is trained by iterating the following process:

1. Get a sample signal $\mathbf{x}$ from the training set and compute its optimal sparse code $\mathbf{z}^*$ as described in the previous section.

2. Keeping $\mathbf{z}^*$ fixed, update $\mathbf{U}$ with one step of stochastic gradient descent: $\mathbf{U} \leftarrow \mathbf{U} - \nu \frac{\partial L_e}{\partial U}$, where L_e is the loss function in (3).

In this paper, we adopt a simple encoder architecture given by:

$$f_e(\mathbf{x}, \mathbf{W}, \mathbf{b}) = h_\theta(\mathbf{Wx} + \mathbf{b}) \quad (4)$$

where $\mathbf{W}$ is a filter matrix, $\mathbf{b}$ is a vector of trainable biases and h_θ is the shrinkage function given by $h_\theta(\mathbf{x})_i =$

$\text{sgn}(x_i)(|x_i| - \theta_i)_+$ (Figure 1). The shrinkage function sets any code components below a certain threshold θ to zero, which helps ensure that the predicted code will be sparse. Training the encoder is done by iterating the above process, with $\mathbf{U} = \{\mathbf{W}, \mathbf{b}, \theta\}$. Note that once the encoder is trained, inferring sparse codes is very efficient, as it essentially requires a single matrix-vector multiplication.

3. LEARNING AUDIO FEATURES

In this section we describe the features learned on music data using PSD.

3.1 Dataset

We used the GTZAN dataset first introduced in [29], which has since been used in several works as a benchmark for the genre recognition task [2, 3, 6, 12, 18, 25]. The dataset consists of 1000 30-second audio clips, each belonging to one of 10 genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock. The classes are balanced so that there are 100 clips from each genre. All clips are sampled at 22050 Hz.

3.2 Preprocessing

To begin with, we divided each clip into short frames of 1024 samples each, corresponding to 46.4ms of audio. There was a 50% overlap between consecutive frames. We then applied a Constant-Q transform (CQT) to each frame, with 96 filters spanning four octaves from C_2 to C_6 at quarter-tone resolution. For this we used the toolbox provided by the authors of [27]. An important property of the CQT is that the center frequencies of the filters are logarithmically spaced, so that consecutive notes in the musical scale are linearly spaced. We then applied subtractive and divisive *local contrast normalization* (LCN) as described in [15], which consisted of two stages. First, from each point in the CQT spectrogram we subtracted the average of its neighborhood along both the time and frequency axes, weighted by a Gaussian window. Each point was then divided by the standard deviation of the new neighborhood, again weighted by a Gaussian window. This enforces competition between neighboring points in the spectrogram, so that low-energy signals are amplified while high-energy ones are muted. The entire process can be seen as a simple form of automatic gain control.

3.3 Features Learned on Frames

We then learned dictionaries on all frames in the dataset, using the process described in 2.2. The dictionary size was set to 512, so as to get overcomplete representations. Once the dictionary was learned, we trained the encoder to predict sparse representations using the process in 2.3. In both cases, we used the Fast Iterative Shrinkage-Thresholding algorithm (FISTA) [1] to compute optimal sparse codes. Some of the learned basis functions are displayed in Figure 2. One can see single notes and what appear to be series of linearly spaced notes, which could correspond to chords, harmonics or harmonies. Note that some of the basis functions appear to be inverted, since the code coefficients can be negative. A number of the learned basis functions also seem to have little recognizable structure.

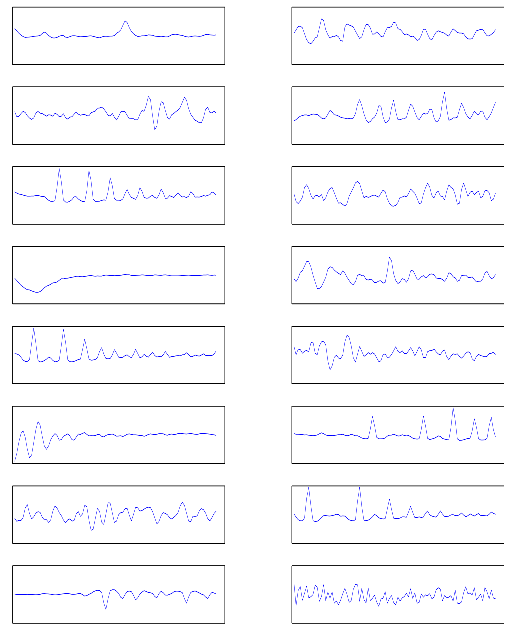

Figure 2. A random subset of the 512 basis functions learned on full CQT frames. The horizontal axis represents log-frequency and ranges from 67 Hz to 1046 Hz.

3.4 Features Learned on Octaves

We also tried learning separate dictionaries on each of the four octaves, in order to capture local frequency patterns. To this end we divided each frame into four octaves, each consisting of 24 channels, and learned dictionaries of size 128 on each one. We then trained four separate encoders to predict the sparse representations for each of the four octaves. Some of the learned basis functions are shown in Figure 3. Interestingly, we find that a number of basis functions correspond to known chords or intervals: minor thirds, perfect fifths, sevenths, major triads, etc. A number of basis functions also appear to be similar versions of each other shifted across frequency. Other functions have their energy spread out across frequency, which could correspond

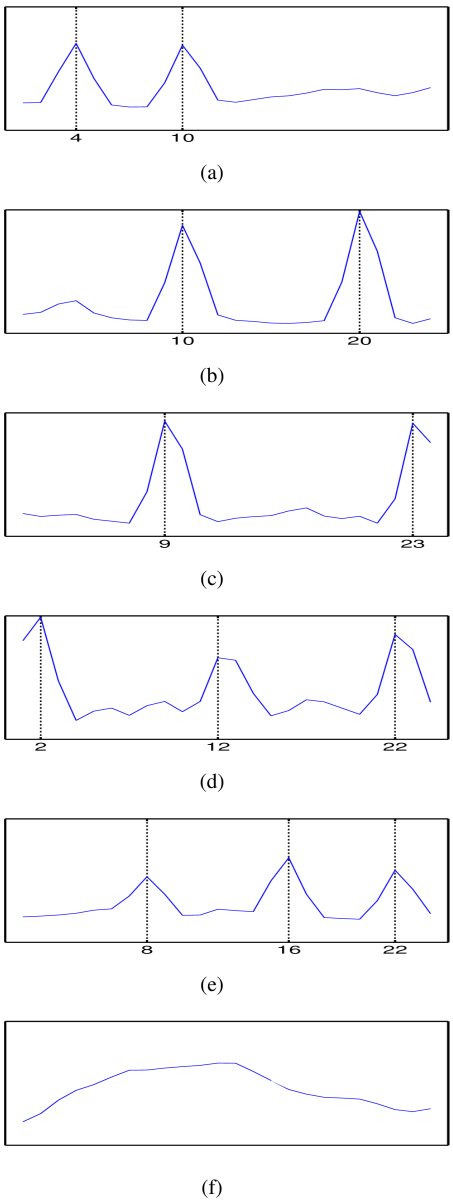

Figure 3. Some of the functions learned on individual octaves. The horizontal axis represents log-frequency. Recall that each octave consists of 24 channels a quarter tone apart. Channel numbers corresponding to peaks are indicated. a) A minor third (two notes 3 semitones apart) b) A perfect fourth (two notes 5 semitones apart) c) A perfect fifth (two notes 7 semitones apart) d) A quartal chord (each note is 5 semitones apart) e) A major triad f) A percussive sound.

to sounds caused by percussive instruments.

3.5 Feature Extraction

Once the dictionaries were learned and the encoders trained to accurately predict sparse codes, we ran all inputs through their respective encoders to obtain their sparse representations using the learned dictionaries. In the case of dictionaries learned on individual octaves, for each frame we concatenated the sparse representations of each of its four octaves, all of length 128, into a single vector of size 512. Extracting sparse features for the entire dataset, which contains over 8 hours of audio, took less than 3 minutes, which shows that this feature extraction system is scalable to industrial-size music databases.

4. CLASSIFICATION USING LEARNED FEATURES

We now describe the results of using our learned features as inputs for genre classification. We used a linear Support Vector Machine (SVM) as a classifier, using the LIBSVM library [5]. Linear SVMs are fast to train and scale well to large datasets, which is an important consideration in MIR.

4.1 Aggregated Features

Several authors have found that aggregating frame-level features over longer time windows substantially improves classification performance [2, 3, 12]. Adopting a similar approach, we computed aggregate features for each song by summing up sparse codes over 5-second time windows overlapping by half. We applied absolute value rectification to the codes beforehand to prevent components of different sign from canceling each other out. Since each sparse code records which dictionary elements are present in a given CQT frame, these aggregate feature vectors can be thought of as histograms recording the number of occurrences of each dictionary element in the time window.

4.2 Classification

To produce predictions for each song, we voted over all aggregate feature vectors in the song and chose the genre with the highest number of votes. Following standard practice, classification performance was measured by 10-fold cross-validation. For each fold, 100 songs were randomly selected to serve as a test set, with the remaining 900 serving as training data. This procedure was repeated 10 times, and the results averaged to produce a final classification accuracy.

Our classification results, along with several other results from the literature, are shown in Figure 4. We see that PSD features learned on individual octaves perform significantly better than those learned on entire frames.[1] Furthermore,

[1] In an effort to capture chords which might be split among two of the octaves, we also tried dividing the frequency range into 7 octaves, overlapping by half, and similarly learning features on each one. However, this did

Classifier	Features	Acc. (%)
CSC	Many features [6]	92.7
SRC	Auditory cortical feat. [25]	92
RBF-SVM	Learned using DBN [12]	84.3
Linear SVM	Learned using PSD on octaves	$\mathbf{83.4 \pm 3.1}$
AdaBoost	Many features [2]	83
Linear SVM	Learned using PSD on frames	$\mathbf{79.4 \pm 2.8}$
SVM	Daubechies Wavelets [19]	78.5
Log. Reg.	Spectral Covariance [3]	77
LDA	MFCC + other [18]	71
Linear SVM	Auditory cortical feat. [25]	70
GMM	MFCC + other [29]	61

Figure 4. Genre recognition accuracy of various algorithms on the GTZAN dataset. Our results with standard deviations are marked in bold.

our approach outperforms many existing systems which use hand-crafted features. The two systems that significantly outperform our own rely on sophisticated classifiers based on sparse representations (SRC) or compressive sampling (CSC). The fact that our method is still able to reach competitive performance while using a simple classifier indicates that the features learned were able to capture useful properties of the audio that distinguish between genres. One possible interpretation is that some of the basis functions depicted in Figure 3 represent chords specific to certain genres. For example, perfect fifths (e.g. power chords) are very common in rock, blues and country, but rare in jazz, whereas quartal chords, which are common in jazz and classical, are seldom found in rock or blues.

4.3 Discussion

Our results show that automatic feature learning is a viable alternative to using hand-crafted features. Our approach performed better than most systems that pair signal processing feature extractors with standard classifiers such as SVMs, Nearest Neighbors or Gaussian Mixture Models. Another positive point is that our feature extraction system is very fast, and the use of a simple linear SVM makes this method viable on any size dataset. Furthermore, the fact that the features are learned in an unsupervised manner means that they are not limited to a particular task, and could be used for other MIR tasks such as chord recognition or autotagging.

We also found that features learned on octaves performed better than features learned on entire frames. This could be due to the fact that in the second case we are learning four times as many parameters as in the first, which could lead to overfitting. Another possibility is that features learned on octaves tend to capture relationships between fundamental notes, whereas features learned on entire frames also seem to capture patterns between fundamentals and their harmonics, which could be less useful for distinguishing between genres.

One aspect that needs mentioning is that since we performed the unsupervised feature learning on the entire dataset (which includes the training and test sets without labels for each of the cross-validation folds), our system is technically akin to "transductive learning". Under this paradigm, test samples are known in advance, and the system is simply asked to produce labels for them. We subsequently conducted a single experiment in which features were learned on the training set only, and obtained an accuracy of 80%. Though less than our overall accuracy, this result is still within the range observed during the 10 different cross-validation experiments, which went from 77% to 87%. The seemingly large deviation in accuracy is likely due to the variation of class distributions between folds.

There are a number of directions in which we would like to extend this work. A first step would be to apply our system to different MIR tasks, such as autotagging. Furthermore, the small size of the GTZAN dataset does not exploit the system's ability to leverage large amounts of data in a tractable amount of time. For this, the Million Song Dataset [4] would be ideal.

A limitation of our system is that it ignores temporal dependencies between frames. A possible remedy would be to learn features on time-frequency patches instead. Preliminary experiments we conducted in this direction did not yield improved results, as many 'learned' basis functions resembled noise. This requires further investigation. We could also try training a second layer of feature extractors on top of the first, since a number of works have demonstrated that using multiple layers can improve classification performance [12, 15, 16].

5. CONCLUSION

In this paper, we have investigated the ability for PSD to automatically learn useful features from constant-Q spectrograms. We found that the features learned capture information about which chords are being played in a particular frame. Furthermore, these learned features can perform at least as well as hand-crafted features for the task of genre recognition. Finally, the system we proposed is fast and uses a simple linear classifier which scales well to large datasets.

In future work, we will apply this method to larger datasets, as well as a wider range of MIR tasks. We will also experiment with different ways of capturing temporal dependencies between frames. Finally, we will investigate using hierarchical systems of feature extractors to learn higher-level features.

not yield an increase in accuracy.

6. REFERENCES

[1] A. Beck and M..Teboulle: "A Fast iterative Shrinkage-Thresholding Algorithm with Application to Wavelet-Based Image Deblurring," *ICASSP '09* , pp. 696-696, 2009.

[2] J. Bergstra, N. Casagrande, D. Erhan, D. Eck and B. Kegl: "Aggregate features and AdaBoost for music classification," *Machine Learning*, 65(2-3):473-484, 2006.

[3] J. Bergstra, M. Mandel and D. Eck: "Scalable genre and tag prediction using spectral covariance," *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010.

[4] T. Bertin-Mahieux, D. Ellis, B. Whitman and P. Lamere: "The million song dataset," *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[5] Chih-Chung Chang and Chih-Jen Lin: "LIBSVM: a library for support vector machines," *software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2001.

[6] K. Chang, J. Jang and C. Iliopoulos: "Music genre classification via compressive sampling," *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 387-392, 2010.

[7] S.S. Chen, D.L. Donoho, and M.A. Saunders: "Atomic Decomposition by Basis Pursuit," *SIAM Journal on Scientific Computing,* 20(1):33-61, 1999

[8] A. Courville, J. Bergstra and Y. Bengio: "A Spike and Slab restricted Bolztmann Machine" *Journal of Machine Learning Research*, W&CP 15, 2011.

[9] I. Daubechies, M. Defrise and C. De Mol : "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint," *Comm. on Pure and Applied Mathematics*, 57:1413-1457, 2004.

[10] D.L. Donoho and M. Elad: "Optimally sparse representation in general (nonorthogonal) dictionaries via L_1 minimization,". *Proceedings of the National Academy of Sciences, USA* , 199(5):2197-2202, 2003 Mar 4.

[11] K. Gregor and Y. LeCun: "Learning Fast Approximations of Sparse Coding," *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

[12] P. Hamel and D. Eck: "Learning Features from Music Audio with Deep Belief Networks," *11th International Society for Music Information Retrieval Conference (ISMIR 2010).* .

[13] P. Herrera-Boyer, G. Peeters and S. Dubnov: "Automatic Classification of Musical Instrument Sounds," *Journal of New Music Research*, vol. 32, num 1, March 2003.

[14] K. Kavukcuoglu, M.A. Ranzato, Y. LeCun: "Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition," *Computational and Biological Learning Laboratory, Technical Report*, CBLL-TR-2008-12-01,

[15] Y. LeCun, K. Kavukvuoglu and C. Farabet: "Convolutional Networks and Applications in Vision," *Proc. International Symposium on Circuits and Systems (ISCAS'10), IEEE, 2010.*

[16] H. Lee, Y. Largman, P. Pham, and A. Ng: "Unsupervised feature learning for audioclassification using convolutional deep belief networks," *Advances in Neural Information Processing Systems (NIPS) 22, 2009*.

[17] H. Lee, A. Battle, R. Raina and A.Y. Ng: "Efficient Sparse Coding Algorithms," *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[18] T. Li and G. Tzanetakis: "Factors in automatic musical genre classification," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 2003.

[19] T. Li, M. Ogihara and Q. Li: "A comparative study on content-based music genre classification," *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*, 2003.

[20] Y. Li and S. Osher: "Coordinate descent optimization for l1 minimization with application to compressed sensing; a greedy algorithm," *Inverse Problems and Imaging* , 3(3):487-503, 2009.

[21] S. Mallat and Z. Zhang: "Matching Pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing,* 41(12):3397:3415, 1993.

[22] P.-A. Manzagol, T. Bertin-Mahieux and D. Eck: "On the use of sparse time relative auditory codes for music," . *In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR), 2008*.

[23] M. Nourouzi, M. Ranjbar and G. Mori: "Stacks of Convolutional Restricted Boltzmann Machines for Shift-Invariant Feature Learning," *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2009.

[24] B. Olshausen and D. Field: "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, 1996.

[25] Y. Panagakis, C. Kotropoulos, and G.R. Arce: "Music genre classification using locality preserving non-negative tensor factorization and sparse representations," *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, pages 249-254, 2009.

[26] G. Peeters: "A large set of audio features for sound description (similarity and classification) in the cuidado project". Technical Report, IRCAM, 2004.

[27] C. Schoerkhuber and A. Klapuri: "Constant-Q transform toolbox for music processing," *7th Sound and Music Computing Conference*, Barcelona, Spain, 2010.

[28] E. Smith and M. Lewicki: "Efficient Auditory Coding" *Nature*, 2006.

[29] G. Tzanetakis and P. Cook: "Musical Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing,* 10(5):293-302, 2002.

[30] Jianchao Yang, Kai Yu, Yihong Gong, Thomas Huang: "Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.