LEARNING SPARSE FEATURE REPRESENTATIONS FOR MUSIC ANNOTATION AND RETRIEVAL

Juhan Nam
CCRMA
Stanford University
juhan@ccrma.stanford.edu

Jorge Herrera
CCRMA
Stanford University
jorgeh@ccrma.stanford.edu

Malcolm Slaney
Yahoo! Research
Stanford University
malcolm@ieee.edu

Julius Smith
CCRMA
Stanford University
jos@ccrma.stanford.edu

ABSTRACT

We present a data-processing pipeline based on sparse feature learning and describe its applications to music annotation and retrieval. Content-based music annotation and retrieval systems process audio starting with features. While commonly used features, such as MFCC, are handcrafted to extract characteristics of the audio in a succinct way, there is increasing interest in learning features automatically from data using unsupervised algorithms. We describe a systemic approach applying feature-learning algorithms to music data, in particular, focusing on a high-dimensional sparse-feature representation. Our experiments show that, using only a linear classifier, the newly learned features produce results on the CAL500 dataset comparable to state-of-the-art music annotation and retrieval systems.

1. INTRODUCTION

Automatic music annotation (a.k.a. music tagging) and retrieval are hot topics in the MIR community, as large collections of music are increasingly available. Therefore, tasks such as music discovery have become progressively harder for humans without the help of computers. Extensive research has been done on these topics [20], [11], [8] [5]. Also, different datasets have become standards to train and evaluate these automatic systems [19], [12].

Training for most automatic systems use audio content—in the form of audio features—as the input data. Traditionally well-known audio features, such as MFCC, chroma and spectral centroid, are used to train algorithms to perform the annotation and retrieval tasks. These "handcrafted" features usually capture partial auditory characteristics in a highly condensed form, ignoring many details of the input data. While such engineered features have proven to be valuable, there is increasing interest in finding a better feature representation by learning from data in an unsupervised manner. Unsupervised learning is usually conducted either by mapping the input data into a high-dimensional sparse space or by means of deep learning. In this paper, we apply high-dimensional sparse feature-learning to short-term audio spectrograms and construct song-level features for music annotation and retrieval.

In summary, the contributions of this paper are as follows:

- We propose a data preprocessing method to make feature-learning algorithms more effective.
- We demonstrate that the feature-learning algorithms capture rich local timbral patterns of music, useful for discrimination.
- We show that song-level features constructed from the local features achieve results comparable to state-of-art algorithms on the CAL500 dataset using only a linear classifier, and furthermore outperform them with a nonlinear classifier.

1.1 Recent Work

Lee et. al. proposed a Convolutional Deep Belief Network (CDBN) and applied it to the audio spectrogram for music genre and artist classification [14]. Dieleman et. al. also employed the CDBN but on engineered features (EchoNest chroma and timbre features) for artist, genre and key recognition tasks [6]. Our approach is similar to these systems in that the input data is taken from multiple audio frames as an image patch and max-pooling is performed for scalable feature-learning. However, we perform feature learning with a high-dimensional single-layer network and the max-pooling separately after learning the features [2]. While this can limit the representational power, it allows faster and simpler training of the learning algorithms.

Henaff et. al. applied a sparse coding algorithm to a single frame of constant-Q transform spectrogram and aggregated them into a segment-level feature for music genre classification [10]. Likewise, Schlter et. al. compared Restricted Boltzmann Machine (RBM), mean-covariance RBM and DBN on similarity-based music classification [17]. Our approach is also similar to these pipelines. However, in our work we provide deeper insight on the learned features by showing how they are semantically relevant. In addition, we investigate the effect of sparsity and max-pooling on the performance.

Finally, Hamel et. al. showed that simply PCA-whitened spectrogram can provide good performance by combining different types of temporal pooling [9]. Our approach is

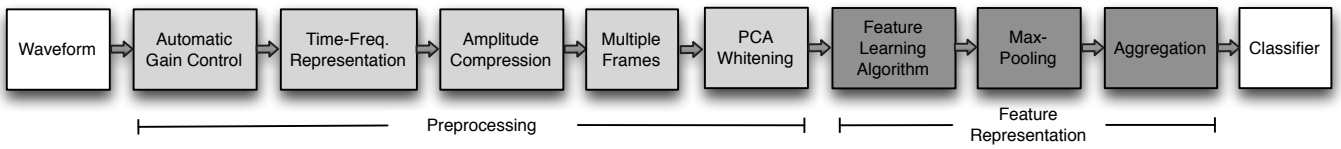

Figure 1: Data processing pipeline for feature representation. This takes an waveform as input and produces a song-level feature for the classifier.

quite different from this work because we encode the PCA-whitened spectrogram into a high-dimensional sparse space and extract features from it.

2. DATA PROCESSING PIPELINE

We perform the music annotation and retrieval tasks using the data processing pipeline shown in Figure 1. Each block in the pipeline is described in this section.

2.1 Preprocessing

Data preprocessing is a very important step to make features invariant to input scale and to reduce dimensionality. We perform several steps of the preprocessing.

2.1.1 Automatic Gain Control

Musical signals are dynamic by nature and each song file in a dataset has different overall volume due to different recording conditions. Thus, we first apply Automatic Gain Control (AGC) to normalize the local energy. In particular, we employ time-frequency AGC using Ellis' method [7]. The AGC first maps FFT magnitude to a small number of sub-bands, computes amplitude envelopes for each band, and uses them to create a time-frequency magnitude envelope over a linear-frequency scale. Then, it divides the original spectrogram by this time-frequency envelope. As a result, the AGC equalizes input signals so they have uniformly distributed spectra over frequency bins.

2.1.2 Time-frequency Representation

A time-frequency representation is an indispensable processing step for musical signals, which are characterized primarily by harmonic or non-harmonic elements. There are many choices of time-frequency representations, each one having different time/frequency resolutions and/or perceptual mappings. In this paper, we chose a mel-frequency spectrogram.

Our initial experiments—based on a spectrogram—showed that using multiple consecutive frames as an input unit for learning algorithms (which is analogous to taking a patch from an image) significantly improves performance over using single frames. However, the FFT size used for musical signals is usually large and thus concatenating multiple frames yields a very high-dimensional vector requiring expensive computation for learning algorithms. Using a moderate number of mel-frequency bins, instead of the straight FFT, preserves the audio content well enough, while significantly reducing the input dimension. We chose 128 mel-frequency bins, following Hamel's work [9], and will present results for various numbers of frames below.

2.1.3 Magnitude Compression

We compress the magnitude using an approximated log scale, $\log_{10}(1 + C|X(t, f)|)$, where $|X(t, f)|$ is the mel-frequency spectrogram and C controls the degree of compression [15]. In general, the linear magnitude of each bin has an exponential distribution. Scaling with a log function gives the magnitude a more Gaussian distribution. This enables the magnitude to be well-fitted with the ensuing PCA whitening, which has an implicit Gaussian assumption.

2.1.4 Multiple Frames

As previously discussed, we take multiple frames as an input unit for feature learning. This approach was used in the convolutional feature-learning algorithm [14]. In that work, however, the multiple frames are taken over the PCA-whitening space where PCA is performed on single frames. In our case, we apply the PCA to multiple consecutive frames for the reasons explained next.

2.1.5 PCA Whitening

PCA whitening is often used as a preprocessing step for independent component analysis or other learning algorithms that capture high-order dependency. It removes pairwise correlation in the input data domain and, as a result, reduces the data dimensionality. Note that the PCA captures short-term temporal correlation as well because it is performed on multiple frames (after vectorizing them).

2.2 Feature Representation

At this point, the input has been processed in a highly reconstructible way so that the underlying structure of the data can be discovered via feature-learning algorithms. In this section, we describe how such algorithms reveal the underlying structure.

2.2.1 Feature Learning Algorithm

We compare three feature-learning algorithms to encode the preprocessed data into high-dimensional feature vectors: K-means clustering, Sparse Coding and Sparse Restricted Boltzmann Machine.

K-means Clustering: K-means clustering learns K centroids from the input data and assigns the membership of a given input to one of the K centroids. In the representational point of view, this can be seen as a linear approximation to the input vectors, $x \approx Ds$, where D is a dictionary

(centroids) and s is an extremely sparse vector that has all zeros but a single "1" that corresponds to the assigned centroid. We use the encoded vector, s, as learned features.

Sparse Coding (SC): Sparse coding is an algorithm to represent input data as a sparse linear combination of elements in a dictionary. The dictionary is learned using the L1-penalized sparse coding formulation. In our experiments, we optimize

$$\min_{D,s^{(i)}} \sum_i \left\| Ds^{(i)} - x^{(i)} \right\|_2^2 + \lambda \left\| s^{(i)} \right\|_1 \quad \text{subject to } \left\| D^{(j)} \right\|_2^2 = 1, \forall j \tag{1}$$

using alternating minimization over the sparse codes $s^{(i)}$, and the dictionary D [3]. We use the absolute value of the sparse code s, as learned features.

Sparse Restricted Boltzmann Machine (sparse RBM): The Restricted Boltzmann Machine is a bipartite undirected graphical model that consists of visible nodes $\mathbf{x}$ and hidden nodes $\mathbf{h}$ [18]. The visible nodes represent input vectors and the hidden nodes represent the features learned by training the RBM. The joint probability for the hidden and visible nodes is defined in Eq. 2 when the visible notes are real-valued Gaussian units and the hidden notes are binary units. The RBM has symmetric connections between the two layers denoted by a weight matrix W, but no connections within hidden nodes or visible nodes. This particular configuration makes it easy to compute the conditional probability distributions, when nodes in either layer is observed (Eq. 3 and 4).

$$-\log P(\mathbf{x},\mathbf{h}) \propto E(\mathbf{x},\mathbf{h}) = \frac{1}{2\sigma^2}\mathbf{x}^T\mathbf{x} - \frac{1}{\sigma^2}\left(\mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h} + \mathbf{h}^T W\mathbf{x}\right) \tag{2}$$

$$p(h_j|\mathbf{x}) = sigmoid(\frac{1}{\sigma^2}(b_j + \mathbf{w}_j^T\mathbf{x})) \tag{3}$$

$$p(x_i|\mathbf{h}) = \mathcal{N}((c_i + \mathbf{w}_i^T\mathbf{h}), \sigma^2), \tag{4}$$

where σ^2 is a scaling factor, $\mathbf{b}$ and $\mathbf{c}$ are bias terms, and W is a weight matrix. The parameters are estimated by maximizing the log-likelihood of the visible nodes. This is performed by block Gibbs sampling between two layers, particularly, using contrastive-divergence learning rule which involves only a single step of Gibbs sampling.

We further regularize this model with sparsity by encouraging each hidden unit to have a pre-determined expected activation using a regularization penalty:

$$\lambda \sum_j (\rho - \frac{1}{m}(\sum_{k=1}^{m} \mathbf{E}[h_j|\mathbf{x}^k]))^2, \tag{5}$$

where $\{\mathbf{x}^1, ..., \mathbf{x}^m\}$ is the training set and ρ determines the target sparsity of the hidden unit activations [13].

Similar to K-means clustering and SC, we can interpret Eq. 4 as approximating input vectors, $\mathbf{x}$, with a linear combination of elements from dictionary W. That is, $\mathbf{x} \approx W\mathbf{h}$ (ignoring the bias term, $\mathbf{c}$). The advantage of RBM over the two algorithms is that the RBM has an explicit encoding scheme, $\mathbf{h} = sigmoid(\frac{1}{\sigma^2}(\mathbf{b} + W^T\mathbf{x})$ from Eq. 3. This enables much faster computation of learned features than SC.

2.2.2 *Pooling and Aggregation*

A song is a very long sequence of data. There are many ways to summarize the data over the entire song. A typical approach to construct a long-term feature is aggregating short-term features by computing statistical summaries over the whole song. However, summarizing all short-term feature over a song dilutes their local discriminative characteristics. Instead, we pool relevant features over smaller segments and then aggregate them by averaging over all the segments in a song.

Since the learned feature vectors are generally sparse and high-dimensional, we performed max-pooling over segments of the song. Max-pooling is an operation that takes the maximum value at each dimension over a pooled area. This is often used in the setting of convolutional neural networks to make features invariant to local transformation. In our experiments, it is used to reduce the smoothing effect of the averaging. In Section 4 we discuss how the pooling size is determined.

2.3 Classification

Music annotation is a multi-labeling problem. We tackle this by using multiple binary classifiers, each predicting the presence of an annotation word. The binary classifier also returns the distance from the decision boundary given a song-level feature. We used the distance as a confidence measure of relevance between a query word and a song for music retrieval.

2.3.1 *Linear SVM*

We use a linear SVM as a reference classifier to evaluate the song-level feature vectors learned by different settings of feature representation. We trained the linear SVM by minimizing the hinge loss given training data. By combining the hinge loss for multiple SVMs as a single objective, we trained them simultaneously, avoiding individual cross-validation for each SVM and thereby saving computation time [16].

2.3.2 *Neural Network*

We also applied a neural network to improve classification performance. For simple evaluation, we used a single hidden layer. However, instead of the cross-entropy, which is usually used as a cost function for a neural network, we employed the hinge loss from the linear SVM above, so that the penalty term is consistent between classifiers. That way, performance difference can be attributed only to the inclusion of the hidden layer.

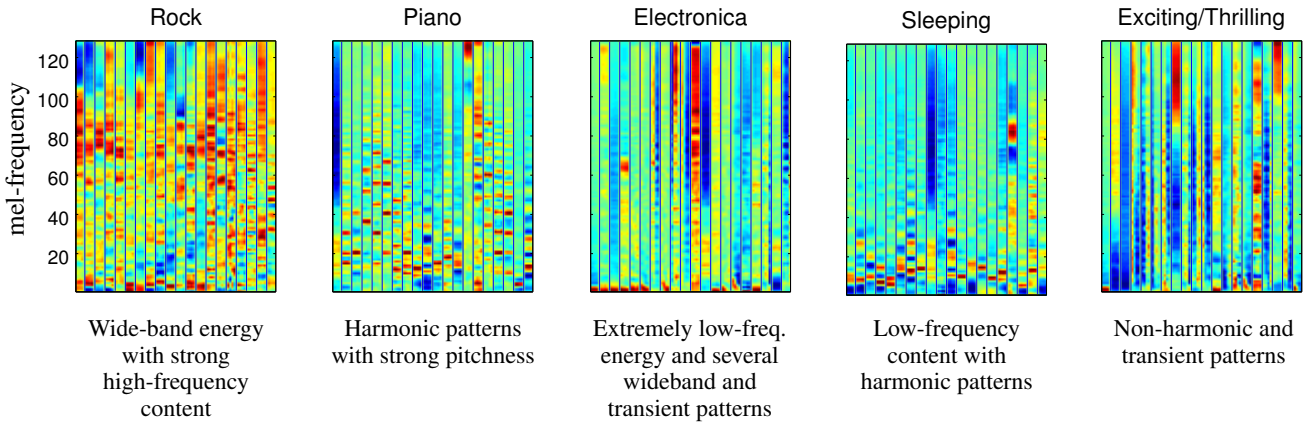

Figure 2: Top 20 most active feature bases (dictionary elements) for five different tags: *Rock*, *Piano*, *Electronica*, *Sleeping* and *Exciting/Thrilling*. Note that all the features come from the same learned dictionary (mel-frequency spectrogram and sparse RBM with 1024 hidden units and 0.01 sparsity), but different types of music use different feature bases.

3. EXPERIMENTS

3.1 Dataset

We evaluated our proposed feature representation on the CAL500 dataset [19]. This dataset contains 502 western songs, each of which was manually annotated with one or more tags out of 174 possibilities, grouped in 6 categories: *Mood*, *Genre*, *Instrument*, *Song*, *Usage*, and *Vocal*. In our experiments, we considered only 97 tags with at least 30 example songs, to be able to compare with results reported elsewhere [20], [4], [8] [5]. In order to apply the full path of our pipeline, we obtained MP3 files of the 502 songs and used the decoded waveforms.

3.2 Preprocessing Parameters

We first resampled the waveform data to 22.05kHz and applied the AGC using 10 sub-bands and attack/delay smoothing the envelope on each band. We computed an FFT with a 46ms Hann window and 50% overlap, which produces a 513 dimensional vector (up to half the sampling rate) for each frame. We then converted it to a mel-frequency spectrogram with 128 bins. In the magnitude compression, C was set to 10 (see section 2.1.3). For PCA whitening and feature learning steps, we sampled 100000 data examples, approximately 200 examples at random positions within each song. Each example is selected as a $128 \times n$ (n=1, 2, 4, 6, 8 and 10) patch from the mel-frequency spectrogram. Using PCA whitening, we reduced the dimensionality of the examples to 49, 80, 141, 202, 263 and 323 for each n by retaining 90% of the variance. Before the whitening, we added 0.01 to the variance for regularization.

3.3 Feature Representation Parameters

We used dictionary size (or hidden layer size) and sparsity (when applicable) as the primary feature-learning parameters. The dictionary size was varied over 128, 256, 512 and 1024. The sparsity parameter was set to $\rho = 0.01$, 0.02, 0.03, 0.05, 0.07 and 0.1 for sparse RBM and $\lambda = 0.5$, 1.0, 1.5 and 2.0 for sparse coding. Max-pooling was performed over segments of length 0.1, 0.25, 0.5, 1, 2, 4, 8, 16, 32 and 64 seconds.

3.4 MFCC

We also evaluated MFCC as a "hand-crafted" feature in order to compare it to our proposed feature representation. Instead of using the MFCC provided from the CAL500 dataset, we computed our own MFCC to match parameters as close as possible to the proposed feature. We used the same AGC and FFT parameters but 40 bins for mel-frequency spectrogram and then applied log and DCT. In addition, we formed a 39-dimensional feature vector by combining its delta and double delta and normalized it by making the MFCC have zero mean and unit variance. The MFCC was also fed into either the classifier directly or the feature-learning step.

3.5 Classifier Parameters

We first subtracted the mean and divided by the standard deviation of each song-level feature in the training set and then trained the classifiers with the features and hard annotation using 5-fold cross-validation. In the neural network, since the classifier is not our main concern, we simply fixed the hidden layer size to 512. After training, we adjusted the distance from the decision boundary using the diversity factor of 1.25, following the heuristic in [11].

4. EVALUATION AND DISCUSSION

4.1 Annotation and Retrieval Performance Metrics

The annotation task was evaluated using Precision, Recall and F-score, following previous work. Precision and Recall were computed based on the methods described by Turnbull [20]. The F-score was computed by first calculating individual F-scores for each tag and then averaging the individual F-scores, similarly to what was done by Ellis [8]. It should be noted that averaging individual F-scores tends to generate lower average F-score than computing the F-score from mean precision and recall values. As for the retrieval, we used the area under the receiver operating characteristic curve (AROC), mean average precision (MAP) and top-10 precision (P10) [8].

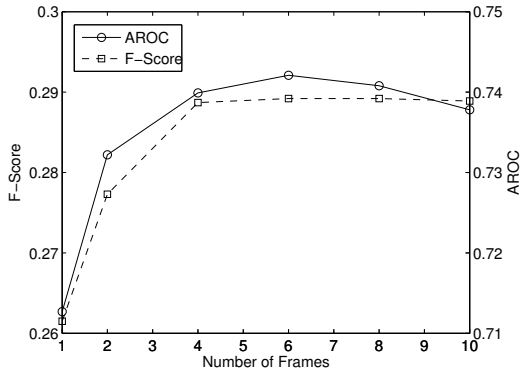

Figure 3: Effect of number of frames (Sparse RBM with 1024 hidden units)

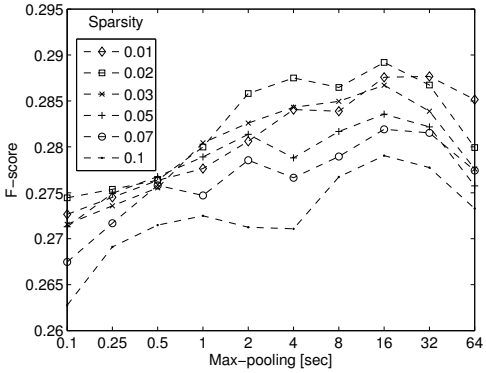

Figure 4: Effect of sparsity and max-pooling (Sparse RBM with 1024 hidden units)

4.2 Visualization

Figure 2 shows most active top-20 feature bases learned on the CAL500 for each tag. They are vividly distinguished by different timbral patterns, such as harmonic/non-harmonic, wide/narrow band, strong low/high-frequency content and steady/transient ones. This indicates the feature learning algorithm effectively maps input data to high-dimensional sparse feature vectors such that the feature vectors (hidden units in RBM) are "selectively" activated by given music.

4.3 Results and Discussion

We discuss the effect of parameters in the pipeline on the annotation and retrieval performance.

4.3.1 Number of Frames

Figure 3 plots F-score and AROC for different number of frames (patch size) taken from the mel-frequency spectrogram. It shows that the performance significantly increases between 1 and 4 frames and then saturates beyond 4 frames. It is interesting that the best results are achieved at 6 frames (about 0.16 second long). We think this is related to the representational power of the algorithm. That is, when the number of frames is small, the algorithm is capable of capturing the variation of input data. However, as the number of frames grows, the algorithm becomes incapable of representing the exponentially increasing variation, in particular, temporal variation.

	Annotation			**Retrieval**		
Data+Algorithm	Prec.	Recall	F-score	AROC	MAP	P10
With AGC						
MFCC only	0.399	0.223	0.242	0.713	0.446	0.467
MFCC+K-means	0.446	0.240	0.270	0.732	0.471	0.492
MFCC+SC	0.437	0.232	0.260	0.713	0.452	0.476
MFCC+SRBM	0.441	0.235	0.263	0.725	0.463	0.485
Mel-Spec+K-means	0.467	0.252	0.287	0.740	0.488	**0.520**
Mel-Spec+SC	0.468	0.252	0.286	0.734	0.482	0.507
Mel-Spec+SRBM	**0.479**	**0.257**	**0.289**	**0.741**	**0.489**	0.513
Without AGC						
MFCC only	0.399	0.222	0.239	0.712	0.444	0.460
MFCC+K-means	0.438	0.237	0.267	0.727	0.465	0.489
Mel-Spec+SRBM	0.449	0.244	0.274	0.727	0.477	0.503

Table 1: Comparison of the performance for different input data and feature learning algorithms. These results are all based on a linear SVM.

4.3.2 Sparsity and max-pooling size

Figure 4 plots F-score for a set of sparsity values and max-pooling sizes. It shows a clear trend that higher accuracy is achieved when the feature vectors are sparse (around 0.02) and max-pooled over segments of about 16 seconds. [1] These results indicate that the best discriminative power in song-level classification is achieved by capturing only a few important features over both timbral and temporal domains.

4.3.3 Input Data, Algorithms and AGC

Table 1 compares the best results on features learned on different types of input data and feature learning algorithms. As shown, the mel-frequency spectrogram significantly outperforms MFCC regardless of the algorithms. Among the feature learning algorithms, K-means and sparse RBM generally perform better than SC. In addition, the results show that the AGC significantly improves both annotation and retrieval performance, regardless of the input features.

4.3.4 Comparison to state-of-the-art algorithms

Table 2 compares our best results to those of state-of-the-art algorithms. They all use MFCC features as input data and represent them either using a Gaussian Mixture Model (GMM), as a bag of frames [20], or Dynamic Texture Mixture (DTM) [4]. They have progressively improved their performance by building on the previous systems, such as, in Bag of Systems (BoS) [8] or Decision Fusion (DF) *decision-fusion*. However, our best system trained with a *linear* SVM shows comparable results. In addition, with nonlinear neural-network classification, our system outperforms the prior algorithms in F-score and all retrieval metrics.

[1] We found that the average length of songs on the CAL500 dataset is approximately 250 seconds, which suggests that aggregating about 16 ($\approx$ 250/16) max-pooled feature vectors over an entire song is an optimal choice.

	Annotation			Retrieval		
Methods	Prec.	Recall	F-score	AROC	MAP	P10
HEM-GMM [20]	0.374	0.205	0.213	0.686	0.417	0.425
HEM-DTM [4]	0.446	0.217	0.264	0.708	0.446	0.460
BoS-DTM-GMM-LR [8]	0.434	**0.272**	0.281	0.748	0.493	0.508
DF-GMM-DTM [5]	**0.484**	0.230	0.291	0.730	0.470	0.487
DF-GMM-BST-DTM [5]	0.456	0.217	0.270	0.731	0.475	0.496
Proposed methods						
Mel-Spec+SRBM+SVM	0.479	0.257	0.289	0.741	0.489	0.513
Mel-Spec+SRBM+NN	0.473	0.258	**0.292**	**0.754**	**0.503**	**0.527**

Table 2: Performance comparison: state-of-the-art (top) and proposed methods (bottom).

5. CONCLUSION AND FUTURE WORK

We have presented a sparse feature representation method based on unsupervised feature-learning. This method was able to effectively capture many timbral patterns of music from minimally pre-processed data. Using a simple linear classifier, our method achieved results comparable to state-of-the-art algorithms for music annotation and retrieval tasks on the CAL500 dataset. Furthermore, our system outperformed them with a non-linear classifier.

To ensure the discriminative power of our proposed feature representation method, we need to evaluate it on larger datasets, such as, the Million Song Dataset [1] or Magnatagatune [12] and also for different classification tasks.

6. REFERENCES

[1] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR*, 2011.

[2] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research*, 2011.

[3] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011.

[4] E. Coviello, A. Chan, and G. Lanckriet. Time series models for semantic music annotation. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

[5] E. Coviello, R. Miotto, and G. Lanckriet. Combining content-based auto-taggers with decision-fusion. In *ISMIR*, 2011.

[6] S. Dieleman, P. Brakel, and B. Schrauwen. Audio-based music classification with a pretrained convolutional network. In *ISMIR*, 2011.

[7] D. Ellis. Time-frequency automatic gain control. web resource, available, `http://labrosa.ee.columbia.edu/matlab/tf_agc/`, 2010.

[8] K. Ellis, E. Coviello, and G. Lanckriet. Semantic annotation and retrieval of music using a bag of systems representation. In *ISMIR*, 2011.

[9] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *ISMIR*, 2011.

[10] H. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun. Unsupervised learning of sparse features for scalable audio classification. In *ISMIR*, 2011.

[11] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *ISMIR*, 2009.

[12] E. Law and L. Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proc. Intl. Conf. on Human factors in computing systems, CHI. ACM*, 2009.

[13] H. Lee, C. Ekanadham, and A. Ng. Sparse deep belief net model for visual area v2. *Advances in Neural Information Processing Systems*, 2007.

[14] H. Lee, P. Pham Y. Largman, and A.Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems*, 2009.

[15] M. Müller, D. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *IEEE Journal on Selected Topics in Signal Processing*, 2011.

[16] J. Nam, J. Ngiam, H. Lee, and M. Slaney. A classification-based polyphonic piano transcription approach using learned feature representation. In *ISMIR*, 2011.

[17] J. Schlter and C. Osendorfer. Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine. In *ICMLA*, 2011.

[18] P. Smolensky. *Information processing in dynamical systems:Foundation of harmony theory*. MIT Press, Cambridge, 1986.

[19] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic description using the CAL500 data set. In *ACM Special Interest Group on Information Retrieval Conference*, 2007.

[20] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 2008.