Towards a More Efficient Sparse Coding Based Audio-word Feature Extraction System

Chin-Chia Michael Yeh and Yi-Hsuan Yang
Research Center for Information Technology Innovation, Academia Sinica, Taiwan.
E-mail: {mcyeh, yang}@citi.sinica.edu.tw Tel: +886-2-2787-2300

***Abstract*—This paper is concerned with the efficiency of sparse coding based audio-word feature extraction system. In particular, we have defined and added the concept of early and late temporal pooling to the classic sparse coding based audio-word feature extraction pipeline, and we have tested them on the genre tags subset of the CAL10k data set. We define temporal pooling as any functions that are able to transforms the input time series representation into a more temporally compact representation. Under this definition, we have examined the following two temporal pooling functions for improving the feature extraction's efficiency, and they are: *Early Texture Window Pooling* and *Multiple Frame Representation*. Early texture window pooling tremendously boost the efficiency by compromising the retrieving accuracy, while multiple frame representation slightly improve both the feature extracting efficiency and retrieving accuracy. Overall, our best feature extraction setup achieves 0.202 in mean average precision on the genre tags subset of the CAL10k data set.**

I. Introduction

As the number of available digital music explosively grows, the importance of research on music information retrieval (MIR) increases. The size of current digital music library has made it difficult to discover new music for human users. One of the countermeasure is to label each song with a set of descriptions (or tags), with which users can search for new music using keyword search. Given that it is impossible to manually label all the music, recent years have witnessed growing interest in developing content-based music tagging (a.k.a. auto-tagging) system [1], [2], [3], [4].

In the early stage of content-based audio analysis for text-based MIR, the song-tag relation is modeled as multi-class problem [5]. Under this multi-class framework, systems are developed specifically to classify music into mutually exclusive groups by genre, emotion, or instrumentation [5], [6], [7]. More recently, for a more comprehensive representation of music, systems which model song-tag relation as multi-class multi-label problem (i.e., auto-tagging) are developed because such multi-class multi-label modeling uses multiple tags to represent a song [1], [3], [8] comparing to the case of using only one tag to represent a song for plain multi-class.

There are generally two types of work targeting auto-tagging system: sophisticated machine learning algorithms with simple features (e.g., MFCC), versus simple classifiers (i.e., binary classifiers) with advanced features. For the first type, Miotto *et al.* [9] considered the tag correlation, Lo *et al.* [10] introduced the idea of cost-sensitive learning, and Coviello *et al.* [11] incorporated temporal information of music. On the contrary, Nam *et al.* [4] and Yang [3] used simple binary classifier with advanced features generated by deep believe net and randomized clustering forest respectively. Out of these two kinds of system, we are particularly interested in the development of later one because simple and efficient classifiers are applicable to large-scale or mobile device applications [3], [12].

Similar to our previous work [3], we are interested in the efficiency of the feature extraction system. Among all the features, audio-word has been shown useful in a variety of MIR tasks [13], [14], [15], [16], owing to its ability to represent music information that happens in a short temporal moment (e.g., "guitar solo") [17]. Based on our previous experience [18], [19], sparse coding based audio-word feature significantly outperforms other conventional coding method, such as Vector Quantization. However, one of the main drawback of sparse coding is the expensive computational cost [3]. As a result, we are interested in improving the efficiency of sparse coding based audio-word feature.

In our previous audio-word extraction system [18], [19], the computational cost of a given song's encoding is proportional to $O(NT)$, where N is the number of frames and T is the average time spend on computing sparse coding for each frame. Based on such relation, we could further decompose the computational cost reduction problem into two subproblems: how to reduce N and how to reduce T? If we look at the low level representation prior to encoding, it usually store as a matrix, where the first dimension is time, and the second dimension is feature. Since sparse coding is performed on a frame by frame fashion, we can see that N is directly linked to the first dimension and T is directly linked to the second dimension. Subsequently, we have looked into the possibility of using different temporal pooling and dimension reduction techniques to reduce the first and second dimension prior to encoding in order to enhance the efficiency of current audio-word feature extraction.

Formally speaking, the term *temporal pooling* is often used to describe functions in which the information from different time is fused together to form a temporally compact representation. For example, a naïve way to generate feature vector for a given song is to temporal pool it by calculating mean and variance across MFCC from all frames [20]. In this study, we have generalized the term to describe any functions that transform the input representation into a more temporally compact representation. In other words, the act of applying any function that reduces the number of instance of a given song can be

called temporal pooling, where the number of instance is the number of frames for frame-based representation (e.g., MFCC) [20] and number of texture-window for texture-window-based representation (e.g., Low-Energy) [5]. Specifically for audio-word framework, since temporal pooling can be applied either prior or after the encoding (e.g., sparse coding), we have define the act of applying temporal pooling before encoding as *early temporal pooling*, and applying temporal pooling afterward as *late temporal pooling*. Both early temporal pooling and late temporal pooling are essential for audio-word features: early temporal pooling has the potential to reduce the computational cost of encoding, and late temporal pooling is a required step for generating the final feature vector for audio-word features.

Although this work represents one of the first attempts that investigate early temporal pooling methods for better *efficiency*, early temporal pooling has been used to improve the *accuracy* of MIR system before [4], [21]. For example, Nam *et al.* [4] incorporated multiple frame representation to increase the performance of their sparse feature, in which the method was performed prior to feature encoding, and Hamel *et al.* [21] designed a multi-scale neural network based auto-tagger by applying various pooling functions between the first two layers. Aside form the performance boost, early temporal pooling techniques could also be viewed as a way to increase the efficiency of any given frame-based feature extraction systems since they reduce the number of frames for each inputed music. However, such "side effect" has never been study before for early temporal pooling methods. We have conducted a preliminary experimentation on the early temporal pooling concept.

In summary, the contributions of this paper include:

- We explore the time/accuracy trade-off of an audio-word feature extraction system when early temporal pooling is applied.
- We demonstrate that our pyramid based bag-of-segments implementation dost not improve the performance while introducing unnecessary computational cost.
- Our best performance system setup achieves the state-of-the-art accuracy of 0.202 in mean average precision on the genre tags subset of the CAL10k data set.

The paper is organized as follows. Section II describes the baseline feature extraction system. Section III describes our enhancement to the baseline system, including the descriptions of multiple frame representation and early texture window pooling. Section IV presents experiment setups and results. Lastly, Section V concludes the paper.

II. Baseline Audio-word Extraction Pipeline

Fundamentally, an audio-word feature extraction pipeline is defined as a function that transform an input music signal's low level representation into higher level audio-word feature. In the baseline system, the input low level representation is first encoded in a frame-by-frame fashion, then the encoded result is temporal pooled and normalized. The detailed explanation of each system components are as follow:

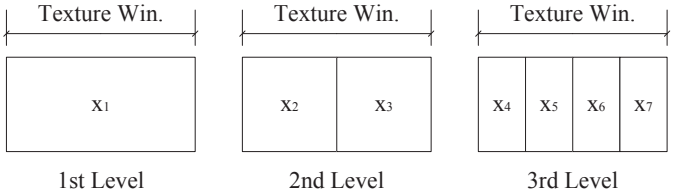

Fig. 1. The three-level pyramid pooling partitioned a given texture window in three different resolutions. Each of the seven partitions is then pooled with summation. The result histograms are concatenated as $x = [x_1, x_2, x_3, \cdots, x_7]^T$ to form the output vector x.

Low Level Representation: Logarithmic scaled 513-D spectrogram with 92ms and 50% overlapping frames is chosen as the low level representation. Refs. [18] and [19] show that spectrogram is the best local representation for sparse coding based audio-word type features comparing to Mel-spectrum, MFCC, Sonogram, and Constant-Q transform.

Encoding: Since our work's target is to improve sparse coding based audio-word feature extraction system, using sparse coding as the encoding method is mandatory. Please refereed to Section II-A for the details of sparse coding.

Late Temporal Pooling: In our previous work [18], [19], [22], we have considered two types of late temporal pooling scheme: *Bag-of-Frames (BoF)* and *Histogram based Bag-of-Segments (HBoS)*. BoF is the most common and simple way to pool audio-word-type features [14], [16], [22], where the audio-words for a given music clip are summed holistically over the whole clip. Alternative methods such as taking maximum or medium for pooling has been found empirically inferior to taking summation [23]. Different in time scale, HBoS sums the audio-words in a segment-by-segment fashion; each segment is defined by the concept of texture window explained in Section II-B. The setting of texture window is 5-second with 50% hop. Additional to BoF and HBoS, we also include a new pooling scheme call *Pyramid based Bag-of-Segments (PBoS)*. Similar to HBoS, the pooling function is applied to each segment. However, rather than pooling sparse coding coefficients for each segment by summation, a pyramid pooling operation is employed [23]. The setting of texture window is also 5-second with 50% hop. Please see Section II-C for details about pyramid pooling. Since the song is represented with multiple segment-based feature vectors when summarized with HBoS and PBoS, while training, all the segments for a given song inherits the song's labels, and the predicted association strength for each given test song is calculated by averaging the predicted association strength over all segments.

Normalization: Based on our finding in [19], [22], normalization is crucial for linear classifier, and power normalized audio-word feature outperformed its correspondent. As we adopted linear SVM as the base classifier for our auto-tagging system, our feature is first power normalized then sum to one normalized. For more detailed information regarding power normalization, please referred to Section II-D.

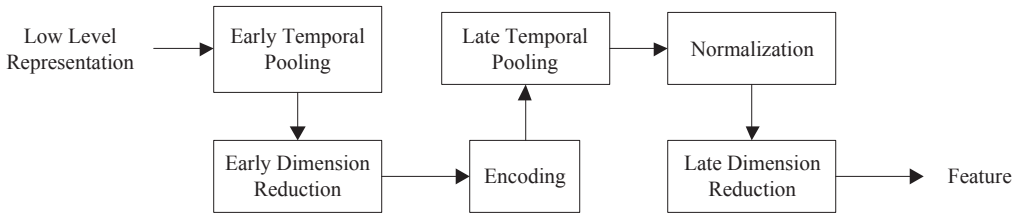

Fig. 2. The audio-word based music signal processing framework.

A. Sparse Coding

Given an input signal vector $x \in \mathbb{R}^m$, the sparse representation problem can be mathematically formulated as

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \frac{1}{2}\|x - D\alpha\|_2^2 + \lambda\|\alpha\|_1, \tag{1}$$

where $\alpha \in \mathbb{R}^k$ is the sparse coding of x, $D \in \mathbb{R}^{m\times k}$ is a given dictionary, and λ is a turning parameter for the trade-off between α's sparsity and the representation accuracy. Typically λ is set to $1/\sqrt{m}$ for that is the classical normalization factor [24], where m is the feature dimension of x. This problem is usually referred to as basis pursuit or Lasso in the machine learning and statistics literature [25]. It can be solved efficiently by off-the-shelf programs such as LARS-lasso [26].

In terms of generating the dictionary for Eq. (1), we employ a first-order stochastic gradient descent algorithm called online dictionary learning (ODL) [24]. ODL is known to be more scalable than standard second-order batch algorithms for its relatively lower computational cost, memory consumption, and capability of learning in an online, rather than batch, fashion. The dictionary learning problem can be formulated as

$$D^* = \underset{D\in\mathcal{C}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\|x_i - D\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1\right), \tag{2}$$

where x_i denotes the i-th signal among a dataset of n signals, and $\mathcal{C} \triangleq \{D \in \mathbb{R}^{m\times k}\}$ is a set of convex matrices in which the l_2 norm of each column d_j is not larger than one, i.e., $d_j^T d_j \leq 1, \forall j$. This constraint is imposed to constrain the energy of the dictionary elements. The formulation in (2) is a joint optimization problem in α and D, and a natural solution is to optimize the two variables in an alternating fashion. We use the implementation due to Mairal*et al.* [24].

B. Texture Window

It has been found that partitioning a song into short segments, each span a number of frames, and generate feature based on the segmentation usually improves the classification accuracy [5], [21], [27]. These segments are called "texture windows" [5] as it should correspond to the minimum time amount of music that is necessary to identify a particular music's timbre, pitch, and loudness.

C. Pyramid Pooling

One of the most popular shift-invariant pooling methods for visual/audio-word-type features is pyramid pooling. Being first proposed in the computer vision community, this method has led to state-of-the-art performances in many tasks [23], [28]. The main idea behind pyramid pooling is to approximate global geometric correspondence in an image by partitioning the image into increasingly fine sub-regions and computing histograms of local features found inside each sub-region. For a three level pyramid, the whole image's features are pooled in the first level. Next, in the second level, the image is divided into 2×2 sub-region, and each sub-region's features are pooled. For the third level, each sub-region is further divided into 2×2 sub-sub-region (i.e., 16 sub-sub-region in total), and features within each sub-sub-region are pooled individually. Finally, all the result histograms are concatenated to form the output feature vector.

The pyramid pooled features are more shift-invariant comparing to directly summed features [28]. Huang *et al.* [23] extended this idea to acoustic event by partition the sound clip into segments similar to the sub-regions partition for image. Unlike images, sounds are 1-D data. Therefore, the partition split the clip into 2 segments instead of 2×2 segments. For music clips, we apply pyramid pooling to each segment as in [23], meaning we partition the music segment into increasingly fine sub-segment and compute histograms of local feature for each sub-segment. We use a three-level pyramid structure as shown in Fig. 1 in this work, and each segment is defined by a 5-second with 50% hop fixed texture window. We name the feature produced by this pooling method *Pyramid based Bag-of-Segments (PBoS)*.

D. Power Normalization

Given an input feature vector $w \in \Re^k$, the power normalization can be calculated with

$$w^* = \operatorname{sign}(w)|w|^a, \tag{3}$$

where $\operatorname{sign}(\cdot)$ is the sign function and $a \in [0, 1]$ is a pre-set parameter, and Jégou *et al.* [29] has empirically determined that $a = 0.5$ constantly leads to near-optimal results. Such transformation has been shown to increase the performance for a BoF based image search system, due to its ability to

reduce the influence of bursty visual elements. It can also be interpreted as variance stabilizing transform, which corrects the dependence between the variance and the mean. It has been applied to BoF, GMM, and Fisher vector, yielding improved performances comparing to their original version in both image and music applications [19], [22], [29].

III. Efficient Audio-word Extraction Pipeline

Fig. 2 shows the complete feature extraction process for this study. The main difference with the baseline process described in Section II is the addition of early temporal pooling, early dimension reduction, and late dimension reduction. The details about each added component is as follow:

Early Temporal Pooling: As we defined in Section I, early temporal pooling is refereed to functions that convert the raw low level representation to a more temporally compact representation. Under this definition, four early temporal pooling functions are discussed detailedly in Section III-A. Within them, we only conduct experiments on early texture window pooling and multiple frames representation due to time limitation.

Early Dimension Reduction: Aside from using early temporal pooling to reduce the number of instance for encoding, we could also use dimension reduction algorithm to reduce the dimension of each instance. Among all the dimension reduction algorithm, we have selected Principle Component Analysis (PCA) due to its popularity. Additionally, the act of applying PCA to low level representation has been known as PCA whitening, and widely used in the literature [4], [21]. Throughout the experiment, the number of principle component is set to maintain 90% of variance.

Late Dimension Reduction: Although this component does not affect the efficiency of feature extraction, we still include this in our evaluation because it is related to the overall system efficiency. Once again, PCA is chosen as the dimension reduction algorithm, and the number of principle component is set to maintain 90% of variance.

A. Early Temporal Pooling

We propose the following four early temporal pooling methods:

Early Texture Window Pooling (ETWP): Utilizing the concept of texture window defined in Section II-B, Each segment is pooled by maximum, mean, and variance functions with fix 3-second texture window. Then, the pooled results from each function are linearly mapped to the range $[0, 1]$, and concatenated together.

Multiple Frames Representation (MFR): It has been shown that when multiple consecutive frames are concatenated and used as the input vectors for feature learning algorithms (e.g., Vector Quantization, Sparse Coding, or Deep Learning) yields significantly superb results comparing to using just a single frame [4]. Ref. [4] also empirically determined the optimal number of frames is 6. As a result, we adopt their best setting, and generate multiple frames representation by concatenating six consecutive spectrum with 50% hop.

Random Down-sampling: Because of the music signals' repetition natural, it is possible to maintain sufficient information for a high level semantic tag while only a randomly selected subset of frames are used in the auto-tagger. Thus, random down-sampling could be thought as an applicable early temporal pooling method.

Clustering-based Down-sampling: Akin to the idea of random down-sampling, Clustering-based down-sampling also chooses a subset of frames of a given song to represent the whole song. But, instead of using randomly selection, Clustering-based down-sampling adopt $k$means algorithm to cluster all the frames of a given song into k cluster, then selects the nearest frame around each centroid as the new set of frames representing the given song.

IV. Experiment

We perform the auto-tagging experiments on genre tag subset of CAL10k data set [8], which consist of 10,870 partially annotated songs from 4,597 artists. Up to 1,053 tags are used to annotate each song, and the song-tag association is defined by expert musicologists of the music service company Pandora[1]. Because the original audio files are not provided, we have collected the 30-second audio previews of 7,799 songs in this data set using the 7digital API.[2] In terms of dictionary learning, we use an external data set, USPOP [30], for better generalizability. USPOP consists of 8,764 tracks from 400 manually selected popular artists.

Following [8], the performance measurements used in this study are the area under the receiver operating characteristic curve (AUC), mean average precision (MAP), 10-precision (P10), and R-precision (PR). The training and test sets are partitioned also based on the specification of Tingle *et al.* [8]. Five different partitions are used, and the average performance across the five partitions is reported.

A. Binary Classifier Based Auto-tagging System

Auto-tagging system is generally designed to solve a multi-label, multi-class problem. That is to say, each song can be labeled with single or multiple tags. One of the simplest ways to attack this problem is breaking down the complex multi-label, multi-class problem into a series of binary classifying problems. For each tag, a binary classifier is constructed to predict whether the tag should be labeled to a given song, and the decision value can be used as an indication on level of association.

We consider two types of feature: BoF-type feature, where the song is represented with a single holistic feature vector, and BoS-type feature, where the song is represented with multiple segment-based feature vectors. For the first type of feature, the feature vector is simply input to linear SVM. For the second type of feature, all the segments for a given song inherits the song's labels while training, and the association strength is calculated by averaging the association strength over all segments for prediction.

[1] http://www.pandora.com
[2] http://developer.7digital.com

TABLE I
THE FEATURE DIMENSION AND NUMBER[A] OF INSTANCE FOR EACH EXPERIMENT SETUP.

Exp. ID	Early Temporal Pooling			Early Dim. Reduction		Late Temporal Pooling			Late Dim. Reduction		Audio-word Feature		Encode Time[b]
	Pooling Method	Feat. Dim.	# of Inst.	Dim. Red.	Feat. Dim.	Pooling Method	Feat. Dim.	# of Inst.	Dim. Red.	Feat. Dim.	Feat. Dim.	# of Inst.	
1	No	–	–	No	–	BoF	1024	1	No	–	1024	1	8.30
2	No	–	–	No	–	BoF	1024	1	Yes	197	197	1	8.30
3	No	–	–	No	–	HBoS	1024	10	No	–	1024	10	8.30
4	No	–	–	No	–	HBoS	1024	10	Yes	347	347	10	8.30
5	No	–	–	No	–	PBoS	7168	10	No	–	7168	10	8.30
6	No	–	–	No	–	PBoS	7168	10	Yes	1562	1562	10	8.30
7	ETWP	1539	19	No	–	BoF	1024	1	No	–	1024	1	0.27
8	ETWP	1539	19	No	–	BoF	1024	1	Yes	497	497	1	0.27
9	ETWP	1539	19	Yes	299	BoF	1024	1	No	–	1024	1	0.15
10	ETWP	1539	19	Yes	299	BoF	1024	1	Yes	524	524	1	0.15
11	MFR	3078	428	No	–	BoF	1024	1	No	–	1024	1	6.15
12	MFR	3078	428	No	–	BoF	1024	1	Yes	155	155	1	6.15
13	MFR	3078	428	No	–	HBoS	1024	10	No	–	1024	10	6.15
14	MFR	3078	428	No	–	HBoS	1024	10	Yes	306	306	10	6.15
15	MFR	3078	428	Yes	1440	BoF	1024	1	No	–	1024	1	5.09
16	MFR	3078	428	Yes	1440	BoF	1024	1	Yes	160	160	1	5.09
17	MFR	3078	428	Yes	1440	HBoS	1024	10	No	–	1024	10	5.09
18	MFR	3078	428	Yes	1440	HBoS	1024	10	Yes	303	303	10	5.09

[a] The numbers here are for a 30 seconds long music clip, and each spectrogram consist of 1290 frames of 513-D spectrum
[b] average time (second) spend on encoding a song

TABLE II
THE RESULT ACCURACIES OF DIFFERENT EXPERIMENT SETUPS

Exp. ID	AUC	MAP	P10	PR	Predict Time [a]
1	0.834	0.171	0.219	0.179	1.08
2	0.836	0.169	0.220	0.179	0.20
3	0.838	0.168	0.212	0.179	41.19
4	0.839	0.169	0.215	0.179	19.10
5	0.837	0.169	0.211	0.179	83.43
6	0.836	0.166	0.209	0.177	81.52
7	0.787	0.118	0.159	0.133	0.77
8	0.795	0.123	0.163	0.136	0.50
9	0.783	0.119	0.158	0.133	0.83
10	0.789	0.122	0.164	0.140	0.58
11	0.846	0.193	0.248	0.206	0.94
12	0.848	0.190	0.242	0.205	0.12
13	0.851	0.197	0.246	0.206	34.77
14	0.850	0.194	0.245	0.206	7.05
15	0.848	0.194	0.251	0.207	0.95
16	0.848	0.189	0.245	0.201	0.12
17	0.854	0.202	0.253	0.214	47.19
18	0.850	0.194	0.245	0.205	7.55

[a] average time (minute) spend on predict association strength for a fold in the five-fold cross validation.

Throughout the experiment, we employ linear SVM from LIBLINEAR [31] to train binary classifiers. The efficiency measurements are done on a server with Intel Xeon CPU E5-2680 @ 2.70GHz processors.

B. Late Temporal Pooling

This section seeks the best way to perform late temporal pooling for efficient audio-word features. To this end, we generate the audio-word feature based on the feature extraction pipeline outlined in Section II. The experiment results are shown in Table I and II, and experiment 1 through 6 are discussed in this section.

Since experiment 1 through 6 share the same low level representation and encoding method, we uses prediction time as the indication of efficiency for each late temporal pooling method. We do not include training time in this discussion because training is considered as an off-line process, and the efficiency of an off-line process is less important. From this set of experiment results, we have three observations: First, the prediction time of PBoS and HBoS is much greater than BoF's, while BoF yield similar accuracy. Thus, it should be fair to say that using PBoS and HBoS in music genre auto-tagging system brings no merit. Second, PBoS's prediction time is double of HBoS's while HBoS's prediction time is 38 times of BoF's without lat dimension reduction. This fact suggests that the number of instance could be more important comparing to feature dimension. Third, although the dimension for the dimension reduced PBoS is much smaller than its original, the difference in prediction time is negligible. Based on above three observations, we can conclude that late dimension reduced BoF is the best option for building a music genre auto-tagger when the prediction time is crucial.

C. Early Temporal Pooling

In this section, we have tested two of the methods we mentioned in Section III-A, and they are ETWP and MFR. Since the goal of adopting them into the system is to reduce the computational cost of encoding, we have use the encoding time as the indication if efficiency. Experiment 7 through 18 in Table I and II are discussed in this section.

First of all, ETWP have dramatically increase the efficiency of encoding with a speed up factor of 55 given that the number of instance have dropped from 1209 to 19. Additionally, comparing to other feature, only ETWP feature's accuracy benefit from late dimension reduction while others' accuracies do not. However, the accuracies of ETWP feature are 5%

TABLE III
OUR BEST RESULT COMPARING TO THE STATE-OF-THE-ART PERFORMANCES

Feature	Classifier	AUC	MAP	P10	PR
Our Best[a]	SVM-L[d]	0.854	0.202	0.253	0.214
Our Fastest[b]		0.789	0.122	0.164	0.140
Our Balanced[c]		0.848	0.189	0.245	0.201
ENT+Δ [8]	GMM[e]	0.887	0.211	0.266	0.224
	SVM-L	0.668	0.051	0.064	0.057
	SVM-R[f]	0.743	0.079	0.104	0.091
	BDS[g]	0.801	0.114	0.157	0.129
SC [3]	SVM-L	0.813	0.128	0.165	0.136
	SVM-HI[h]	0.812	0.153	0.200	0.170
RCFT [3]	SVM-L	0.807	0.130	0.167	0.144
	SVM-HI	0.773	0.121	0.167	0.133

[a] Experiment 17
[b] Experiment 10
[c] Experiment 16
[d] Linear SVM
[e] GMM with Mixutre Hierarchies EM algorithm
[f] Kernelized SVM with RBF kernel
[g] Boosted decision stumps
[h] Kernelized SVM with Histogram Intersection kernel

worse than the baseline features. On the other hand, the speedup factor (1.63) of MFR is much smaller comparing to ETWP, and it could caused by that the difference in number of instance between MFR and low level representation is also much smaller than ETWP. In spite of that, MFR audio-word feature outperforms the baseline feature by 2%, and MFR feature with HBoS surpass BoF by a small margin. Thus, MFR slightly enhance both the result accuracy and encoding efficiency. Overall, we can conclude that ETWP is a viable solution for systems that requires efficient feature extraction system such as large-scale or mobile device application, while MFR is favorable for feature extraction systems that are less strict on efficiency.

According to Table III, although we are unable to get the full set of 10,870 audio files, roughly speaking, our best (i.e., Experiment 17) and balanced (i.e., Experiment 16) feature are only behind [8], which uses GMM-based classifier. If we only consider results with SVM-L-based classifier, both feature extraction systems are superior comparing to others. In consequence, it should be fair to say that our features are more distinguishable in linear space. On the other hand, our fastest feature (i.e., Experiment 10) is on par with other features when SVM-L-based classifier is used. This shows that even if our modification has traded 5% of accuracy for efficiency, our feature still remain competitive to the EchoNest feature used in [8] and randomized clustering forest feature used in [3].

V. CONCLUSIONS

In this paper, we have evaluated the ideas of early/late temporal pooling and early/late dimension reduction for improving the efficiency of sparse coding based audio-word feature extraction system. Specifically, for early temporal pooling, we have tested the idea of early texture window pooling (ETWP) and multiple frames representation (MFR). Our evaluation on genre-based auto-tagging on the CAL10k dataset shows that ETWP is a promising method, as it trades 5% of accuracy for a speedup factor of 55. On the other hand, although the speed gained from MFR is only marginal, MFR is able to enhance the accuracy by a small margin. Our experiments lead to three setups 'best,' 'balanced,' and 'fastest' that can be used depending on whether accuracy or efficiency is of higher concern. With the 'best' setup, we are able to obtain MAP 0.202 for genre-based auto-tagging on CAL10k, one of the best results when linear SVM is employed as the classifier. With the 'fastest' setup, we are able to reduce the encoding time from 5.09 sec/song to 0.15 sec/song, while attaining MAP that is on par with some other existing efficient autotaggers. For future work, we plan to implement the random down-sampling and clustering-based down-sampling methods for early temporal pooling, and to design better temporal pooling mechanics for sparse coding based audio-word feature based on the findings in this paper.

ACKNOWLEDGMENT

This work was supported by the National Science Council of Taiwan under Grants NSC 101-2221-E-001-017, NSC 102-2221-E-001-004-MY3 and the Academia Sinica Career Development Award.

REFERENCES

[1] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," in *IEEE TASLP*, 2008.
[2] M. Levy and M. Sandler, "Music information retrieval using social tags and audio," in *IEEE TMM*, 2009.
[3] Y.-H. Yang, "Towards real-time music auto-tagging using sparse features," in *ICME*, 2013.
[4] J. Nam, J. Herrera, M. Slaney, and J. Smith, "Learning sparse feature representations for music annotation and retrieval," in *ISMIR*, 2012, pp. 565–560.
[5] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Processing*, 2002.
[6] Y.-H. Yang and H. H. Chen, "Machine recognition of music emotion: A review," in *ACM TIST*, 2012.
[7] S. Scholler and H. Purwins, "Sparse approximations for drum sound classification," *IEEE J. Sel. Topics Signal Processing*, vol. 5, no. 50, pp. 933–940, 2011.
[8] D. Tingle, Y. E. Kim, and D. Turnbull, "Exploring automatic music annotation with "acoustically-objective" tags," in *ISMIR*, 2010.
[9] R. Miotto and G. Lanckriet, "A generative context model for semantic music annotation and retrieval," in *IEEE TASLP*, 2012.
[10] H.-Y. Lo, J.-C. Wang, H.-M. Wang, and S.-D. Lin, "Cost-sensitive multi-label learning for audio tag annotation and retrieval," *IEEE TMM*, vol. 13, no. 3, pp. 518–529, 2011.
[11] E. Coviello, A. B. Chan, and G. Lanckriet, "Time series models for semantic music annotation," in *IEEE TASLP*, 2011.
[12] J.-C. Wang, H.-M. Wang, and S.-K. Jeng, "Playing with tagging: A real-time music tagging player," in *ICASSP*, 2012.
[13] J.-C. Wang, H.-S. Lee, H.-M. Wang, and S.-K. Jeng, "Learning the similarity of audio music in bag-of-frames representation from tagged music data," in *ISMIR*, 2011.
[14] B. McFee, L. Barrington, and G. R. G. Lanckriet, "Learning content similarity for music recommendation," *IEEE Trans. Audio, Speech and Lang. Processing*, vol. 20, no. 8, 2012.
[15] J. Wülfing and M. Riedmiller, "Unsupervised learning of local features for music classification," in *ISMIR*, 2012, pp. 139–144.
[16] J.-Y. Liu, C.-C. M. Yeh, Y.-C. Teng, and Y.-H. Yang, "Bilingual analysis of song lyrics and audio words," in *ACM Multimedia*, 2012, pp. 829–832.
[17] M. I. Mandel, D. Eck, and Y. Bengio, "Learning tags that vary within a song," in *ISMIR*, 2010, pp. 399–404.

[18] C.-C. M. Yeh and Y.-H. Yang, "Supervised dictionary learning for music genre classification," in *ACM ICMR*, 2012.
[19] L. Su, C.-C. M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang, "A systematic evaluation of the bag-of-frames representation for music information retrieval," in *IEEE TMM*, 2013.
[20] A. Meng, P. Ahrendt, J. Larsen, and L. K. Hansen, "Temporal feature integration for music genre classification," *IEEE Trans. Audio, Speech and Lang. Processing*, vol. 15, no. 5, pp. 1654–1664, 2007.
[21] 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, pp. 729–734.
[22] C.-C. M. Yeh, L. Su, and Y.-H. Yang, "Dual-layer bag-of-frames model for music genre classification," in *ICASSP*, 2013.
[23] P.-S. Huang, J. Yang, M. Hasegawa-Johnson, F. Liang, and T. S. Huang, "Pooling robust shift-invariant sparse representations of acoustic signals," in *Interspeech*, 2012.
[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *ICML*, 2009, pp. 689–696.
[25] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Computing*, vol. 20, pp. 33–61, 1998.
[26] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, 2004.
[27] J. Bergstra and B. Kegl, "Aggregate features and adaboost for music classification," in *Machine Learning*, 2006, vol. 65, pp. 473–484.
[28] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.
[29] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE TPAMI*, 2012.
[30] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman, "A large-scale evaluation of acoustic and subjective music similarity measures," in *Computer Music Journal*, 2003.
[31] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Machine Learning Research*, 2008.