UNSUPERVISED LEARNING OF LOCAL FEATURES FOR MUSIC CLASSIFICATION

Jan Wülfing and Martin Riedmiller
University of Freiburg, Germany
{wuelfj,riedmiller}@tf.uni-freiburg.de

ABSTRACT

In this work we investigate the applicability of unsupervised feature learning methods to the task of automatic genre prediction of music pieces. More specifically we evaluate a framework that recently has been successfully used to recognize objects in images. We first extract local patches from the time-frequency transformed audio signal, which are then pre-processed and used for unsupervised learning of an overcomplete dictionary of local features. For learning we either use a bootstrapped k-means clustering approach or select features randomly. We further extract feature responses in a convolutional manner and train a linear SVM for classification. We extensively evaluate the approach on the GTZAN dataset, emphasizing the influence of important design choices such as dimensionality reduction, pooling and patch dimension on the classification accuracy. We show that convolutional extraction of local feature responses is crucial to reach high performance. Furthermore we find that using this approach, simple and fast learning techniques such as k-means or randomly selected features are competitive with previously published results which also learn features from audio signals.

1. INTRODUCTION

Automatic categorization of music pieces into categories such as mood, artist or genre is a widely studied topic in music information retrieval. Those categorization tasks basically consist of two steps: feature selection/extraction and classification. Designing and selecting good features for a certain task is demanding and requires expert knowledge about the domain at hand. Nonetheless, a wide range of those hand designed features have been proposed in the past. More recently there has been a growing interest in methods that automatically learn features from data in an unsupervised fashion. Those methods have been very successful on a range of recognition benchmarks for images as well as audio data (see Section 2).

In this work, we investigate the applicability of a k-means based unsupervised feature learning framework that has been used successfully for object recognition in RGB-D images [1] to the problem of genre classification of music pieces.

This framework allows us to fast and flexibly learn local features of different sizes and shapes and to show whether features that span the whole frequency range, only parts of it or even features that cover several consecutive points in time perform best for the task of genre prediction.

To do so, we need to transform the raw audio signal into the time-frequency domain, for which researches have used varying transformations in the past. We determine the influence of this choice by evaluating the feature learning on two different transformations.

After learning an overcomplete dictionary of local features, we extract feature responses in a convolutional manner. Although computationally more expensive, we show that convolutional extraction, together with the right pooling scheme, improves recognition performance significantly.

The paper is structured as follows: In Section 2 we give a short review of approaches that also learn features from audio data, followed by a description of the learning framework in Section 3 and 4. We extensively evaluate the parameters of the learning framework and show its potential on the GTZAN dataset [14] by reporting competitive results in Section 5.

2. RELATED WORK

There is a range of feature learning methods that have been used to tackle music information retrieval tasks e.g. sparse coding [6], principal component analysis [5], deep belief networks [4,8], or a mean-covariance restricted Boltzmann machine [11].

All those feature learning frameworks rely on a transformation of the raw audio signal to the spectral domain. Frequently used is the short time Fourier transformation (with varying window lengths), which can be mel-frequency scaled [5, 11]. Henaff et al. [6] apply the constant-Q transform [12].

Features learned by those approaches differ in size and shape, e.g. some approaches rely on features that cover single time frames [4–6], only parts of the frequency range [6] or even learn time-frequency features that span several consecutive points in time [11].

Learning feature codebooks using simple k-means has a long tradition and has also been applied to audio tasks in [11, 13].

However, Coates et al. [3] just recently found that good image features can be learned using k-means if state-of-the-art image pre-processing and feature encoding is used. This finding could be confirmed and extended for RGB-D images by Blum et al. [1] who used a convolutional bootstrapped k-means procedure to successfully learn RGB-D features for object recognition in “3D” images.

3. LEARNING FEATURE RESPONSES

Our goal is to learn a set of feature responses $D \in \mathbb{R}^{N \times k}$ given a set of input vectors $X = \{x^{(1)}, \dots, x^{(m)}\}$ with $x^{(i)} \in \mathbb{R}^N$. The input vectors are patches of size $v \times w$ extracted from a training set represented as column vectors. Each value is represented using d channels (e.g. with RGB images $d = 3$, with spectrograms $d = 1$) and hence $N = v \cdot w \cdot d$. Random patches of size $v \times w$ are extracted to build the training set X. Once X is known we apply a pre-processing step followed by the unsupervised learning algorithm.

3.1 Pre-processing

As a pre-processing step we first normalize all patches contained in X by subtracting their mean and dividing by the standard deviation. Afterwards a whitening transformation [7] is applied to the patches. The purpose of the whitening transformation is to ensure that values are decorrelated and have unit variance. This step is crucial to ensure a good quality of the learned feature responses as shown in [3]. We use PCA whitening, which allows us to drop insignificant dimensions from the input data. This results in increased feature extraction speed and can improve feature quality as shown in [1]. If dimensionality reduction is used, we chose to keep the first n components thereby projecting each extracted patch $x \in \mathbb{R}^N$ to a lower dimensional vector $x' \in \mathbb{R}^n$.

3.2 Unsupervised learning

We use a k-means approach to learn k centroids building the feature response dictionary D by clustering the extracted patches X. Although k-means is a very simple unsupervised learning algorithm that is easy to implement, it has recently been shown that it is competitive to other unsupervised learning algorithms when learning local, low-level features from pre-processed image data [3]. Apart from its simplicity the main advantage of using k-means over other algorithms is that it is very fast and scales well to a large amount of centroids. It can therefore be trivially parallelized on current computer hardware in a map-reduce manner and allows us to learn large, over-complete feature dictionaries that can be expensive to learn using other unsupervised learning approaches.

3.2.1 Bootstrapping

To further improve upon the feature quality that can be achieved using standard k-means, as well as the required run time until convergence, we use a bootstrapping learning scheme as proposed in [1] to train the k centroids.

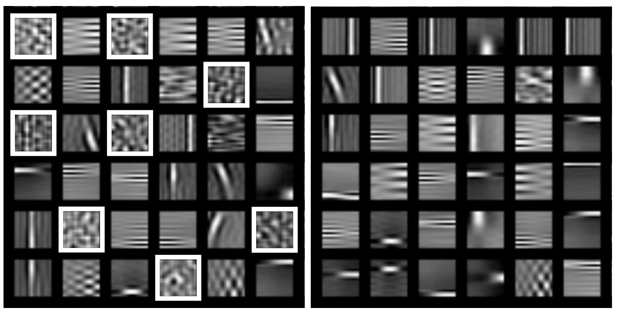

Figure 1: Comparison of 16×16 features learned on the GTZAN dataset using the CQT transform without and with bootstrapping. (a) Without bootstrapping several cluster centers, marked in white, do not represent good feature responses due to the high dimensional space in which k-means clustering is performed. (b) When bootstrapping is enabled all learned centroids correspond to nicely localized features.

We first cluster in the subspace spanned by the first p principal components and fill the learned centroids with zeros for all other $n - p$ dimensions. These centroids are then used to *warm start* the clustering procedure in the n dimensional PCA whitened space.

Without bootstrapping some features are badly localized, which is an artifact of clustering in a high dimensional space (e.g. 256 dimensions if patches of size 16×16 are used). This effect is visible in Fig. 1 where affected features are marked white. When the bootstrapping procedure is used the features are well distributed over the whole feature space by pre-training the centroids on the major principal components. The consecutive clustering procedure in the complete feature space is thus simplified and the badly localized features disappear.

4. FEATURE EXTRACTION

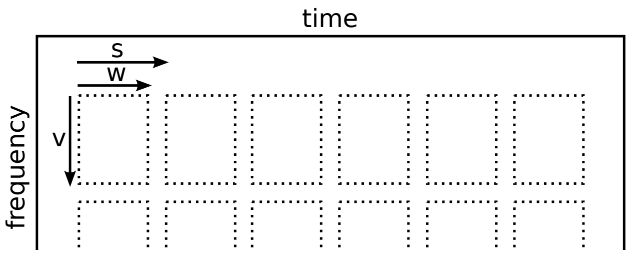

Figure 2: Schematic of the convolutional extraction scheme. Note that with a stride s smaller than v or w, extracted patches overlap.

After learning the dictionary, feature responses are extracted from the input data. Employing a convolutional extraction scheme (see Fig. 2), we traverse the input data with stride s and extract patches at all possible positions. Instead of using standard hard k-means where the feature response $f(x)$ is a sparse vector indicating the closest centroid

$$f_i(x) = \begin{cases} 1 & \text{if } c^i = \arg\min_{c^i \in D} \|c^i - x\| \\ 0 & \text{else} \end{cases}, \quad (1)$$

we compute the triangular response to maximize the information content of each feature response. It keeps the information about the distance of the current patch to all centroids $c^i \in D$ that are closer than the average distance $\mu(z) = \frac{1}{k}\sum_{i=1}^{k} z_i$ where $z \in \mathbb{R}^k$ with $z_i = \|c^i - x\|$. In this case $f(x)$ can be defined as

$$f_i(x) = \max(0, \mu(z) - z_i). \quad (2)$$

4.1 Pooling

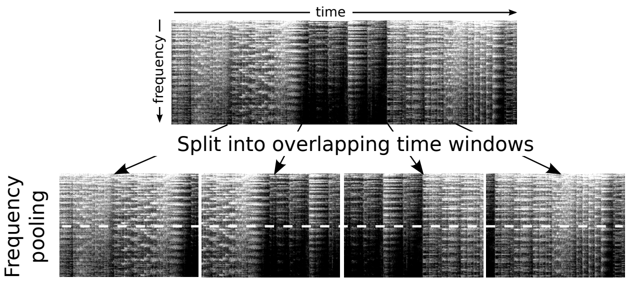

Figure 3: Illustration of the pooling scheme. In the first step, the time-frequency transformed audio signal is split up into overlapping windows. For each window features are extracted and pooled.

Since using all feature responses for classification is computationally expensive - we get a response of size k for each patch that we extract convolutionally - it is common practice to use a pooling scheme to reduce the dimensionality of the feature vector. The term pooling here refers to placing a grid with c cells on the input data and computing a function (e.g. maximum or average) over all feature responses that fall into a grid cell. The dimension of the resulting feature vector is reduced to $c \times k$. For object recognition in images a frequently used grid structure is 2×2. This is a reasonable choice for object recognition tasks where objects are positioned at the center of the image. Here the grid helps to roughly encode the spatial properties of the presented objects in the resulting feature vector. Analogously for audio data, using more than one pool on the time axis results in an encoding of the temporal properties of the feature response. This however is not desired for the task of genre prediction, since characteristic patterns for a certain genre might not always occur at exactly the same time. Additionally we might have to predict the genre based only on a fragment of the song (as it is the case for the GTZAN dataset), underlining the problem that encoding timing may not help, but in fact impair the quality of our prediction. Invariance to timing can be achieved by pooling only once on the time axis (e.g. 2×1). This however may reduce the information content of the feature vector too drastically. To overcome this problem, we split the input data into overlapping time windows of a certain length (similar to [6] and [5]), compute feature responses and pool on each window separately. Each window then serves as input to the classifier and the final result is determined by voting over the classification results of all windows. An illustration of this scheme is depicted in Fig. 3.

5. EVALUATION

We evaluate the performance of the learned features on the GTZAN dataset [14].

5.1 Experimental Setup

5.1.1 Dataset

The GTZAN dataset is organized into 10 distinct genres: Blues, Classical, Country, Disco, Hip hop, Jazz, Metal, Pop, Reggae and Rock. Each genre is represented by 100 song fragments of 30 seconds length.

5.1.2 Pre-processing of audio data

There are several transformations used in the literature to transform the raw audio signal into the time-frequency domain (see Section 2). To determine the influence of this pre-processing choice, we evaluate the feature learning for two different transformations. We apply a short time Fourier transform, calculated on 1024 samples with 512 samples overlap (STFT) and, in a second setting, use the Constant Q-Transform (CQT) [12] spanning 8 octaves, using 64 bins per octave, to create spectrograms of the audio signal. Both spectrograms have exactly the same number of values on the frequency axis (512 values). We also sub sample the CQT to have exactly the same time resolution as the STFT (1292 time frames). This way any possible advantage due to a larger representation can be ruled out.

5.1.3 Classification

For classification we use a linear SVM in a 10-fold cross validation setting.

5.2 Patch dimension and learning techniques

In a first experiment, we show the influence of varying patch sizes and learning techniques on classification accuracy. We learned features using k-means with bootstrapped and randomly initialized cluster centers (chosen at random from the input data). We ran k-means until convergence.

Table 1: Influence of varying patch dimensions and learning methods on classification accuracy. Results are averaged over ten runs of 10-fold cross validation to minimize the influence of random partitioning. The standard deviations are all well below 1%.

Patch size	k-means (random)		k-means (boot.)	
(freq. × time)	STFT	CQT	STFT	CQT
64×1	64.81	70.91	64.72	70.57
128×1	59.59	67.14	68.48	72.19
256×1	65.79	62.12	67.86	70.8
512×1	62.37	66.54	67.69	67.26
16×16	75.2	74.2	75.11	77.7

Features span parts of the frequency axis (64×1, 128×1, 256×1), the whole frequency range (512×1) or frequency and time (16×16). Note that features are extracted convolutionally (with stride 1) if possible which excludes features of size 512×1. In contrast to the smaller patches those features cannot benefit from introducing several pools on the frequency axis, they already span the whole frequency range. That is why we only use one pool in this experimental condition. If not mentioned otherwise, we learn dictionaries of size 800 using PCA whitening and keeping as many components necessary to explain 95% of the variance. Table 1 shows the results of this experiment.

For a small patch size of 64×1 the performance of k-means and bootstrapped k-means is almost equal. Here bootstrapping k-means is unnecessary, since in low dimensions random initialization of the cluster centers suffices. With growing feature size however (e.g. 256×1), random k-means suffers from the effect depicted in Fig. 1, where parts of the dictionary are wasted on ill localized features. Bootstrapping k-means reduces the impact of this problem and affects classification accuracy significantly (e.g. 5.05% improvement with a feature size of 128×1).

Comparing the performance of varying feature sizes, we find that learning features on the whole frequency range (without convolution) has the lowest accuracy, compared to smaller frequency patches. The 16×16 time-frequency features outperform any other setting.

In all settings the STFT accuracies are worse than the results on the CQT transformed audio signal. In addition to the advantages of the Constant-Q transform over the discrete Fourier transform described in [12], we suspect that this is due to the fact that the Constant-Q transform is much sparser and less noisy than the STFT and thereby facilitates learning of good features.

5.3 Pooling and time windows

In this experiment we evaluate the parameters of the pooling scheme described in Section 4.1 used for feature extraction. We employ average pooling in all experiments and vary the number of pools on the frequency axis. We perform experiments on the CQT transformed data. The results for features of size 256×1 and 16×16 (note that both settings share the same number of components) learned with bootstrapped k-means are are shown in Fig. 4a). In all tested settings, increasing the number of frequency pools helps to improve the classification accuracy. Best results are achieved using two to four pools.

In Fig. 4b) the results of varying the length of the time windows are depicted. Accuracy increases with shorter time windows. Depending on the patch size, the optimum is reached with a window length of 1 second (16×16) or 2 seconds (256×1). This finding is in agreement with [5].

5.4 PCA dimensionality reduction and dictionary size

We show the effect of varying the number of principal components kept in Fig. 4c). In the previous experiments, we used exactly as many principal components needed to explain 95 % of the variance, which translates to keeping

Classifier	Features	Accuracy (%)
Linear SVM	Convolutional K-means (16×16) (our)	$\mathbf{85.25 \pm 3.5}$
RBF SVM	DBN [4]	84.3
Linear SVM	PSD on octaves [6]	83.4 ± 3.1
Linear SVM	Convolutional K-means (128×1) (our)	$\mathbf{83.37 \pm 2.54}$
Linear SVM	PSD on frames [6]	79.4 ± 2.8

Table 2: Our results (in bold) compared to previously published results that learn features on the GTZAN dataset. We report the averaged accuracy and standard deviation after one run of 10-fold cross validation.

88 principal components in case of the 16×16 features and 133 for the 256×1 features. We find that for both feature sizes the highest accuracy can be achieved by setting the number of principal components to 100.

Finally, we evaluate the effect of varying the size of the dictionary learned. In Fig. 4d) the results of varying this number are depicted. Increasing the size of the dictionary steadily improves recognition performance.

5.5 Overall performance

To compare our results with previously published results on the GTZAN dataset we learned 1600 features, used 4 frequency pools, time windows of 2 seconds length and kept the first 100 (16×16) and 72 (128×1) principal components. Results are shown in Table 2. With features that span time and frequency, we reach the best result on the GTZAN dataset compared with other approaches that learn features from audio data. There are however approaches that do not learn features in an unsupervised fashion, but focus on sophisticated classifiers and significantly outperform our results (92.7% [2] , 92.4% [9]).

5.6 Additional experiment using random features

Recently randomly selected features were found to perform well on object recognition benchmarks. Saxe et al. [10] attribute the success of random features to the convolutional pooling architecture they are used in. To investigate the role of convolutional feature extraction for audio data, we performed a similar, additional experiment. We chose the same parameters as described in Section 5.5, but instead of learning features, we randomly selected PCA whitened patches without any further clustering and used these as features. Indeed, we found that classification accuracy did not suffer significantly ($85.09\% \pm 3.56$), but the interpretability of the features is lost. This result underlines the importance of convolutional feature extraction.

6. DISCUSSION

Our experiments indicate that convolutional extraction of local feature responses is a viable approach to increase recognition accuracy for the task of genre prediction.

With convolutional extraction there is a trade off between computational complexity and accuracy. Extracting

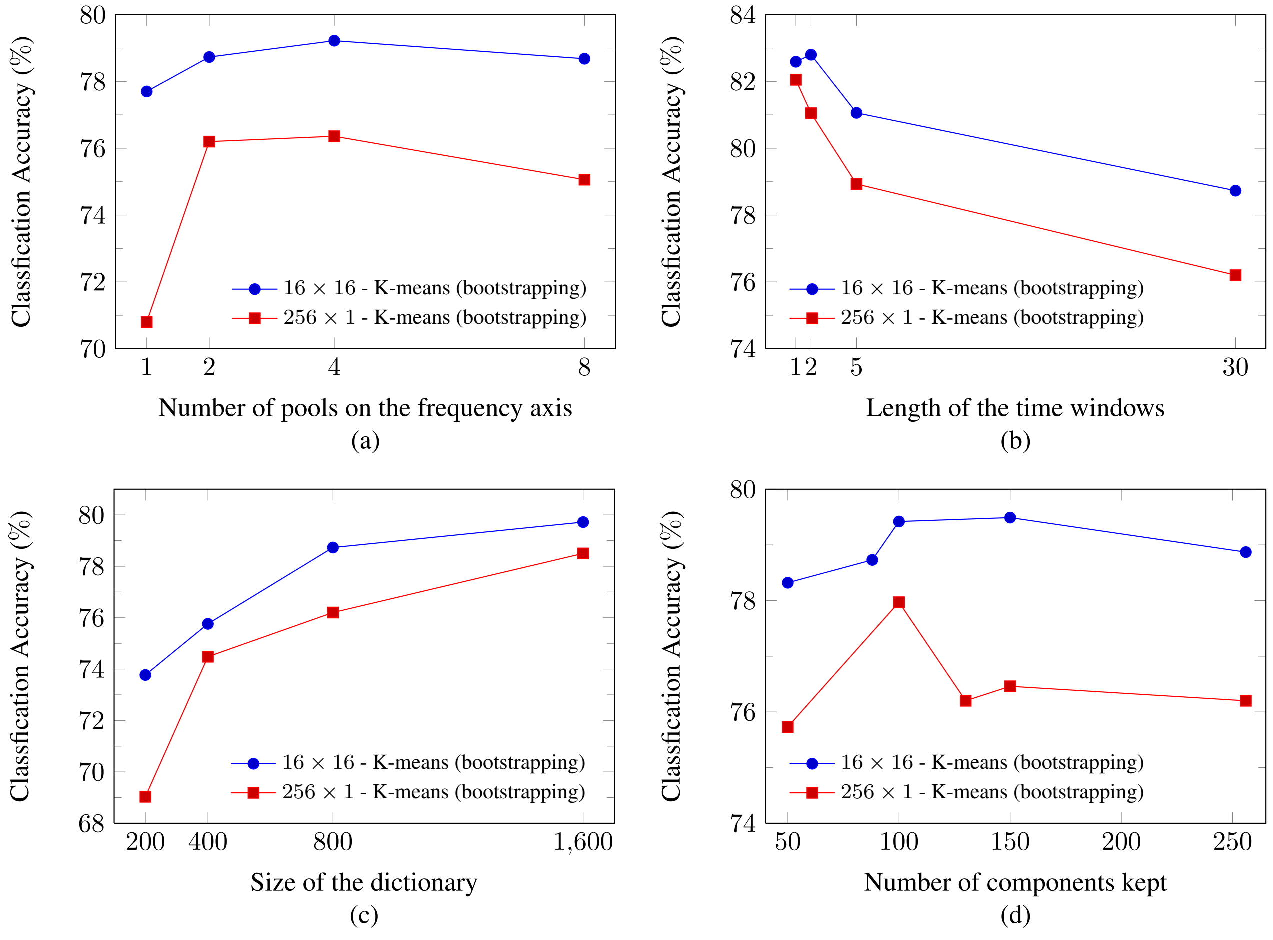

Figure 4: Classification accuracies averaged over ten runs of 10-fold cross validation with (a) varying number of frequency pools, (b) varying time window lengths, (c) varying dictionary size and (d) varying number of components kept.

features convolutionally with a stride of 1 is computationally more expensive than extracting non-overlapping feature responses. To speed up the feature extraction we tried to reduce the size of the spectrograms to a quarter of their original size (128 × 323), which helps twofold. For one, the number of patches that need to be extracted decreases and we are able to learn smaller patch dimensions, which speeds up finding the closest centroids. We found that accuracy did only decrease marginally to $84.77\% \pm 2.6$, when learning patches of size 8×8 on the smaller input. Another way of speeding up the extraction is to increase the stride s. This however has a stronger effect on accuracy, which reduces to $83.45\% \pm 3.3$ (16×16, same setting as in Section 5.5) with a stride of 4.

Another important finding is that time-frequency features perform better in terms of accuracy than frame level features. Nonetheless, features that span the whole frequency range have the advantage of being easily interpretable in musical terms (see Fig. 5 for exemplary features learned only on the frequency axis). This is not the case for local time-frequency patches since the exact frequency is not encoded in those patches. They do however represent patterns of energy distribution over time that can occur at any frequency, e.g. energy remaining constant at one frequency

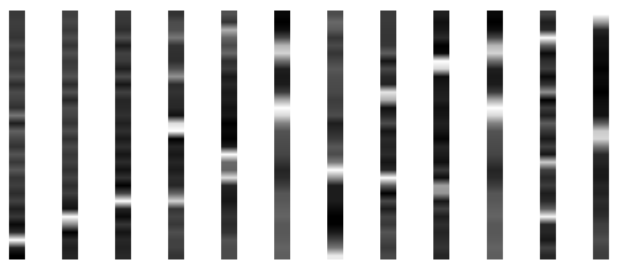

Figure 5: Example features learned on frequency patches 32 × 1 (enlarged for better visualization).

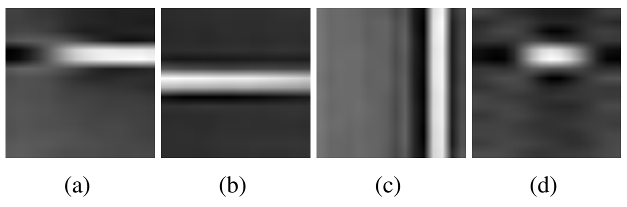

Figure 6: Examples of learned time-frequency features (enlarged for better visualization).

(Fig. 6b), energy spreading across frequencies (Fig. 6c) and note onsets (Fig. 6a and d).

Finally, we show the confusion matrix of the result that was achieved with our best performing features in Fig. 3. Genres that have a low confusion rate include classic, jazz and metal, problematic are rock and pop songs. We believe that the confusion patterns that occur are plausible, e.g. confusing metal with rock songs is a reasonable mistake, since both genres are closely related.

	Bl	Ro	Di	Hi	Ja	Re	Po	Co	Cl	Me
Bl	827	61	1	9	0	20	12	48	0	4
Ro	12	650	55	25	7	25	51	49	0	32
Di	31	36	853	11	2	36	45	12	0	9
Hi	8	5	21	866	0	32	33	0	0	0
Ja	19	6	0	4	961	10	0	12	9	0
Re	48	16	10	22	0	824	15	20	0	0
Po	0	36	33	41	0	18	775	24	0	0
Co	45	110	18	1	0	35	16	830	0	7
Cl	0	5	9	0	30	0	10	1	991	0
Me	10	75	0	21	0	0	43	4	0	948

Table 3: The confusion matrix for ten runs of 10-fold cross validation using features of size 16×16. Genres that have a low confusion rate include classic, jazz and metal, problematic are rock and pop songs.

7. CONCLUSION

In this work, we have presented an approach that predicts the genre of a music piece. We have shown that learning local features using simple and fast techniques like k-means or even randomly selected features is competitive with other more complex learning approaches, if features are extracted convolutionally. We found that time-frequency patches perform better than one dimensional frequency patches and that they reach the highest accuracy to date compared with other learned features on the GTZAN dataset. Furthermore, we have shown that features learned on the CQT transformed audio signal perform better than those learned on the STFT spectrogram. We consider as interesting future work to apply the feature learning to different tasks in the domain of music information retrieval, e.g. auto tagging and also to investigate the possibility of learning a deeper representation on top of the low level features learned so far.

8. REFERENCES

[1] Manuel Blum, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller. A Learned Feature Descriptor for Object Recognition in RGB-D Data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[2] Kaichun K. Chang, Jyh-Shing Roger Jang, and Costas S. Iliopoulos. Music genre classification via compressive sampling. In *ISMIR*, pages 387–392. International Society for Music Information Retrieval, 2010.

[3] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[4] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 339–344, August 2010.

[5] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *In Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR11)*, 2011.

[6] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. Unsupervised Learning of Sparse Features for Scalable Audio Classification. In *Proceedings of the International Society for Music Information Retrieval*, 2011.

[7] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–30, 2000.

[8] Honglak Lee, Yan Largman, Peter Pham, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22*, pages 1096–1104, 2009.

[9] Yannis Panagakis, Constantine Kotropoulos, and Gonzalo R. Arce. Music genre classification using locality preserving non-negative tensor factorization and sparse representations. In *ISMIR*, pages 249–254. International Society for Music Information Retrieval, 2009.

[10] Andrew Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1089–1096, June 2011.

[11] Jan Schlüter and Christian Osendorfer. Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine. In *Proceedings of the 10th International Conference on Machine Learning and Applications (ICMLA 2011)*, 2011.

[12] A. Schoerkhuber, C. and Klapuri. Constant-Q transform toolbox for music processing. In *7th Sound and Music Computing*, 2010.

[13] Klaus Seyerlehner, Gerhard Widmer, and Peter Knees. Frame level audio similarity - a codebook approach. In *Proc. of the 11th International Conference on Digital Audio Effects (DAFx-08)*, 2008.

[14] George Tzanetakis and Perry Cook. Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.