MULTISCALE APPROACHES TO MUSIC AUDIO FEATURE LEARNING

Sander Dieleman and Benjamin Schrauwen
Electronics and Information Systems department, Ghent University
{sander.dieleman, benjamin.schrauwen}@ugent.be

ABSTRACT

Content-based music information retrieval tasks are typically solved with a two-stage approach: features are extracted from music audio signals, and are then used as input to a regressor or classifier. These features can be engineered or learned from data. Although the former approach was dominant in the past, feature learning has started to receive more attention from the MIR community in recent years. Recent results in feature learning indicate that simple algorithms such as K-means can be very effective, sometimes surpassing more complicated approaches based on restricted Boltzmann machines, autoencoders or sparse coding. Furthermore, there has been increased interest in multiscale representations of music audio recently. Such representations are more versatile because music audio exhibits structure on multiple timescales, which are relevant for different MIR tasks to varying degrees. We develop and compare three approaches to multiscale audio feature learning using the spherical K-means algorithm. We evaluate them in an automatic tagging task and a similarity metric learning task on the Magnatagatune dataset.

1. INTRODUCTION

Content-based *music information retrieval* (MIR) techniques can be used to solve a variety of different problems, such as genre classification, artist recognition and music recommendation. They typically have a two-stage architecture: first, features are extracted from music audio signals to transform them into a more meaningful representation. These features are then used as input to a regressor or a classifier, which is trained to perform the task at hand.

Although machine learning techniques such as support vector machines and neural networks have traditionally been popular for the second stage, the features extracted from the audio are typically engineered rather than learned. Feature engineering is a complex and time-consuming process. Furthermore, these features are usually designed with a particular task in mind and are likely not optimally suited for other tasks. A prime example of this are the popular *mel-frequency cepstral coefficients* (MFCCs), which were originally designed for speech processing. MFCCs mainly encode timbre and discard pitch information, which is often undesirable when working with music audio. Despite this, they are still very commonly used for this purpose.

In recent years, feature learning has started to receive more attention from the MIR community. This can be attributed at least in part to a surge of interest in feature learning in general, ever since the emergence of *deep learning* in the mid-2000s [12]. Simply put, the idea of deep learning is to learn a hierarchy of features, organized in layers that correspond to different levels of abstraction. Higher-level features are defined in terms of lower-level features.

A similar evolution has taken place in computer vision and speech processing, where approaches based on deep learning are now commonplace, after improving on the previous state of the art by a large margin [6, 14]. In light of this, Humphrey et al. [13] advocate the use of deep architectures to solve MIR problems. However, recent results in feature learning indicate that simple, shallow (single-layer) feature learning techniques such as the K-means algorithm can also be quite competitive, sometimes surpassing more complicated approaches based on restricted Boltzmann machines, autoencoders and sparse coding [5]. These models are typically much more difficult to tune than the K-means algorithm, which only has one parameter (the number of means). They are often an order of magnitude slower to train as well.

Another recent development in MIR research is the increased interest in *multiscale architectures* [1,8,10]. Music audio exhibits structure on many different timescales: at the lowest level, signal periodicity gives rise to pitch. Periodicity at longer timescales emerges from rhythmic structure, repeated motifs and musical form. These timescales are relevant for different tasks to varying degrees.

Some researchers have explored architectures that are both deep and multiscale: for example, in convolutional neural networks, subsampling layers are often inserted between the convolutional layers, so that the features at each successive level take a larger part of the input into account [7, 16]. Indeed, it is not unreasonable to assume that more complex features will typically span longer timescales. However, these concepts need not necessarily be intertwined; for example in multiresolution deep belief networks [19], both hierarchies are separated.

In this paper, we endeavor to build a versatile feature learning architecture for music audio that operates on multiple timescales, using the spherical K-means algorithm. We compare three multiscale architectures and evaluate the resulting features in an automatic tagging task and a similarity metric learning task on the Magnatagatune dataset

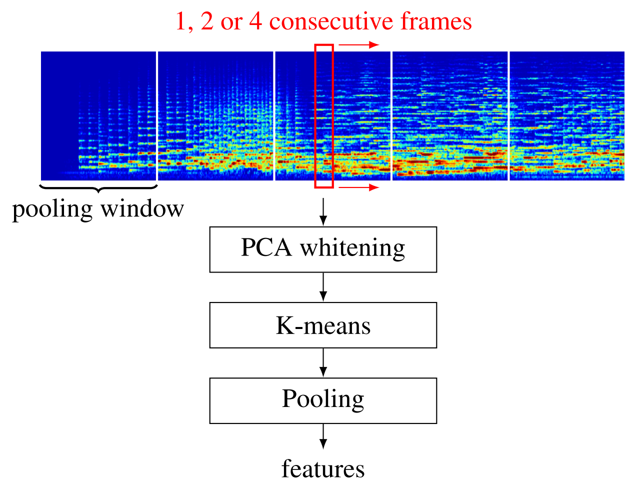


Figure 1: Schematic overview of the feature extraction process at a single timescale. The mel-spectrogram is divided into large pooling windows. Smaller windows consisting of 1, 2 or 4 consecutive frames are extracted convolutionally and PCA-whitened, and then K-means features are extracted. The features are pooled by taking the maximum across time over the pooling windows.

[15]. The rest of the paper is structured as follows: in Section 2 we outline the feature learning algorithm we used. A description of the multiscale architectures we investigated follows in Section 3. Our experiments and evaluation methods are detailed in Section 4 and the results are discussed in Section 5.

2. FEATURE LEARNING

To learn features from music audio, it is typically converted into a time-frequency representation first, such as a mel-spectrogram (see Section 4.1 for details). Feature learning algorithms can then be applied to individual spectrogram frames or windows of consecutive frames. We used the following feature extraction pipeline, which is visualized schematically in Figure 1: first, the mel-spectrograms are divided into large pooling windows, several seconds in length. Smaller windows consisting of 1, 2 or 4 consecutive frames are then extracted convolutionally and PCA-whitened, and K-means features are extracted from the whitened windows. The features are pooled by taking the maximum across time over the pooling windows. Each part of the pipeline is decribed in more detail below. For now, we will assume a typical single-timescale setting. In Section 3, we extend our approach to multiscale time-frequency representations.

2.1 PCA whitening

First, we randomly sample a set of windows from the data, and whiten them with PCA. We keep enough components to retain 99% of the variance. The whitened windows are then used to learn the dictionary. It has been shown that this whitening step considerably improves the features learned by the K-means algorithm [5].

2.2 Spherical K-means

We then use spherical K-means to learn features from the whitened windows. The spherical K-means algorithm differs from more traditional variants in the fact that the means are constrained to have a unit L2 norm (they must lie on the unit sphere). This is achieved by adding a normalization step in every iteration after the means are updated. For a detailed overview of the algorithm, we refer to Coates et al. [4].

K-means has often been used as a dictionary learning algorithm in the past, but it has only recently been shown to be competitive with more advanced techniques such as sparse coding. The one-of-K coding (i.e., each example being assigned to a single mean) is beneficial during learning, but it turns out to be less suitable for the encoding phase [3]. By replacing the encoding procedure, the features become significantly more useful. For spherical K-means, it turns out that using a linear encoding scheme works well: the feature representation of a data point is obtained by multiplying it with the dictionary matrix. To extract features from mel-spectrograms with a sliding window, we have to whiten the windows and extract features, which can be implemented as a single convolution with the product of the whitening matrix and the dictionary matrix.

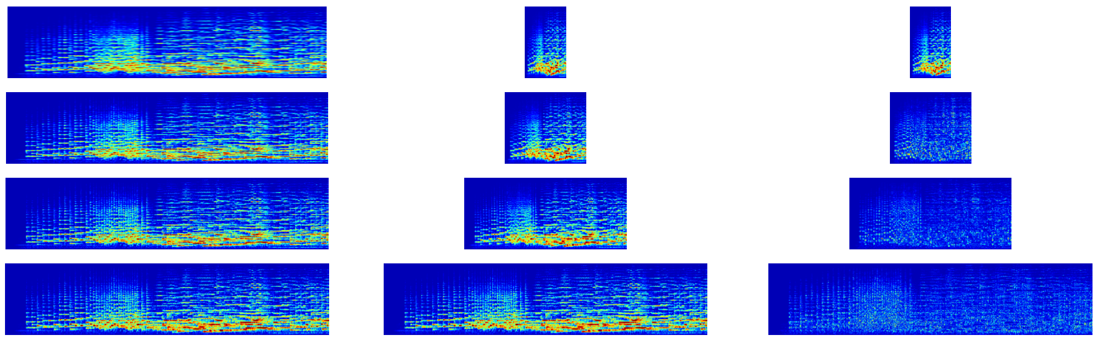
We can also skip the K-means step altogether and use the PCA components obtained after whitening as features directly. These features were referred to as principal mel-spectrum components (PMSCs) by Hamel et al. [11]. They are in fact very similar to MFCCs: if the windows consist of single frames, replacing the PCA whitening step with a discrete cosine transform (DCT) results in MFCC vectors. Both transformations serve to decorrelate the input.

2.3 Pooling

The representation we obtain by extracting features convolutionally from mel-spectrograms is not yet suitable as input to a regressor or classifier. It needs to be summarized over the time dimension first. We pool the features across large time windows several seconds in length. Although a combination of the mean, variance, minimum and maximum across time has been found to work well for this purpose [11], we use only the maximum, because it was found to be the best performing single pooling function. This reduces the size of the feature representation fourfold, which speeds up experiments significantly while having only a limited impact on performance. We used non-overlapping pooling windows for convenience, but our approach does not preclude the use of overlapping windows. Figure 1 shows a schematic overview of the feature extraction process.

3. MULTISCALE APPROACHES

We explore three approaches to obtain multiscale time-frequency representations from mel-spectrograms: multiresolution spectrograms, Gaussian pyramids and Laplacian pyramids [2]. An example of each is given in Figure



(a) Multiresolution spectrograms (b) Gaussian pyramid (c) Laplacian pyramid

Figure 2: Three different 4-level multiscale time-frequency representations of music audio. Level 0 is at the bottom, higher levels correspond to coarser timescales. This figure is best viewed in color.

2. To arrive at a multiscale feature representation, we simply apply the pipeline described in Section 2 to each level separately, and concatenate the resulting feature vectors.

3.1 Multiresolution spectrograms

The most straightforward approach to learning music audio features at multiple timescales is to extract mel-spectrograms with different window sizes. This approach was previously explored by Hamel et al. [10]. For each consecutive level, we simply double the spectrogram window size of the previous level. Although we could also double the hop size to reduce the size of the higher level representations, this was found to decrease performance considerably, so it is kept the same for all levels. Figure 2a shows an example of a set of multiresolution spectrograms.

3.2 Gaussian pyramid

The Gaussian pyramid is a very popular multiscale representation in image processing. Each consecutive level of a Gaussian pyramid is obtained by smoothing and then subsampling the previous level by a factor of 2 in the time dimension. To our knowledge, it has not previously been applied to time-frequency representations of music audio. In this case, mel-spectrograms are extracted only at the finest timescale and all higher levels can be obtained from them. Higher-level representations in a Gaussian pyramid will be smaller in size because of the subsampling step. This means that the pooling windows used in our feature extraction pipeline must be shrunk accordingly. An example of a Gaussian pyramid is shown in Figure 2b. Note that the lowest level of the Gaussian pyramid is identical to that of the multiresolution spectrograms.

3.3 Laplacian pyramid

The Laplacian pyramid can be derived from the Gaussian pyramid by taking each level and subtracting an upsampled version of the level above it. The top level remains the same. The result is that the representations at finer timescales will not contain any information that is already represented at coarser timescales. This reduces redundancy between the levels of the pyramid. An example of a Laplacian pyramid is shown in Figure 2c.

3.4 Modeling local temporal structure

Frames taken from multiresolution spectrograms will automatically model longer-range temporal structure at higher levels, because the spectrogram window size is increased. For the Gaussian and Laplacian pyramids however, this is not the case: any temporal structure present at longer timescales is lost by the downsampling operation that is required to construct the higher levels of the pyramid. Although frames at higher levels of the pyramid are affected by a larger region of the input, they do not reflect temporal structure within this region. To allow for this structure to be taken into account by the feature learning algorithm, we can instead use windows of a small number of consecutive frames as input. This is not useful for multiresolution spectrograms because the temporal resolution is the same at all levels, i.e. higher levels do not have coarser temporal resolutions as is the case for the pyramid representations.

Figure 3 shows a random selection of features learned from windows of 4 consecutive mel-spectrogram frames at different levels in a 6-level Gaussian pyramid, with PCA whitening (top) and with spherical K-means (bottom). Some features are stationary across the time dimension, others resemble percussive events and changes in pitch. Many of the K-means features seem to reflect harmonic structure. This is especially the case for level 1 where the features span roughly half a second, which is around the average duration of a musical note. This type of structure is less pronounced in the PCA features.

4. EXPERIMENTS

4.1 Dataset

We used the Magnatagatune dataset [15], which contains 25863 29-second audio clips taken from songs by 230 artists sampled at 16 kHz, along with metadata and tags. It comes in 16 parts, of which we used the first 12 for training, the 13th for validation and the remaining 3 for testing. We extracted log-scaled mel-spectrograms with 200 components, with a window size of 1024 frames (corresponding to 64 ms) and a hop size of 512 frames (32 ms). For the multiresolution spectrogram representation, the spectrogram window size was doubled for each con-

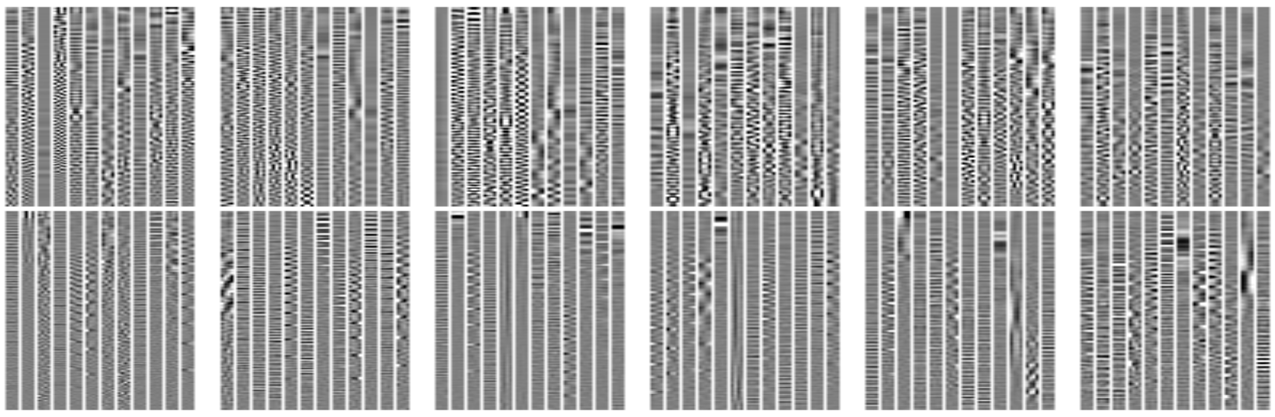


Figure 3: A random selection of features learned with PCA whitening (top) and spherical K-means (bottom) from windows of 4 consecutive mel-spectrogram frames, at different levels in a Gaussian pyramid. The frequency increases from bottom to top.

secutive level. We used 6 levels for all three multiscale representations, numbered 0 to 5. This means that frames at the highest level span 2 seconds and are spaced 1 second apart. We used pooling windows of 128 spectrogram frames at the lowest level (about 4 seconds).

4.2 Tag prediction

The main task we used to evaluate the proposed feature learning architectures is a tag prediction task. This allows us to evaluate the versatility of the representations, because tags describe a variety of different aspects of the music: genre, presence (or absence) of an instrument, tempo and dynamics, among others. All clips in the dataset are annotated with tags from a set of 188. We only used the 50 most frequently occurring tags for our experiments, to ensure that enough training data is available for each of them.

We trained a multilayer perceptron (MLP) on the proposed multiscale feature representations to predict the presence of the tags, with a hidden layer consisting of 1000 rectified linear units [17]. We used minibatch gradient descent with weight decay to minimize the cross entropy between the predictions and the true tag assignments, and stopped training when the performance on the validation set was no longer increasing. Hyperparameters such as the learning rate and the weight decay constant were optimized by performing a random search on the validation set. We trained the MLP on feature vectors obtained from pooling windows. Tag predictions for an entire clip were computed by taking the average of the predictions across all pooling windows. We computed the area under the ROC-curve (AUC) for all tags individually and then took the average across all tags to get a measure of the predictive performance of the trained models.

We evaluated four different feature learning setups: PCA whitening, and spherical K-means with 200, 500 and 1000 means. We also compared 7 different multiscale approaches: Gaussian and Laplacian pyramids with windows of 1, 2 and 4 consecutive frames, and multiresolution spectrograms. This yields 28 different architectures in total.

4.3 Similarity metric learning

We also used the features to learn an audio-based music similarity metric, to further assess their versatility. Using Neighborhood Components Analysis (NCA) [9], we learned a linear projection of the features into a 50-dimensional space, such that similar clips are close together in terms of Euclidean distance. Each clip is mapped into this space by projecting the feature vectors corresponding to each pooling window and then taking the mean across all pooling windows. The linear projection matrix is then optimized with minibatch gradient descent to project clips by a given artist into the same region. This approach to learning a music similarity metric was previously explored by Slaney et al. [18].

NCA is based on a probabilistic version of K-nearest neighbor classification, where neighbors are selected probabilistically proportionally to their distance and each data point inherits the class of its selected neighbor. The objective is then to maximize the probability of correct classification. We report this probability on the test set.

5. RESULTS

5.1 Architectures

The results for the tag prediction task obtained with each of the 28 different architectures are shown in Figure 4. All reported results are averaged across 10 MLP training runs with different initializations. Unfortunately we cannot directly compare our results with those of Hamel et al. [10], because they used a different version of the dataset.

The first thing to note is that using features learned with spherical K-means almost always yields increased performance, although the difference between using 500 or 1000 means is usually small. Interestingly, the best performing architecture uses a Laplacian pyramid, with features learned from single frames. This is somewhat unexpected, because it implies that grouping consecutive frames into windows so that the feature learning algorithm can capture temporal structure is not necessary for this type of multi-

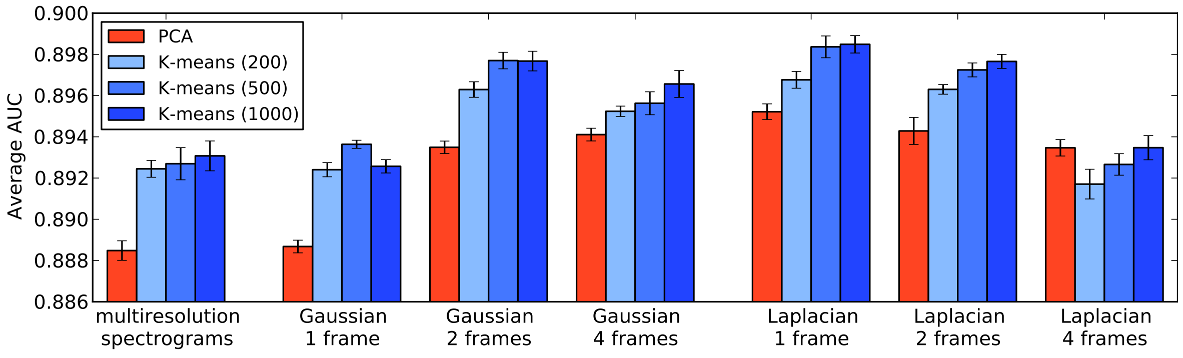


Figure 4: Results for the tag prediction task, for 28 different multiscale feature learning architectures. All reported results are averaged across 10 MLP training runs with different initializations. Error bars indicate the standard deviation across these 10 runs.

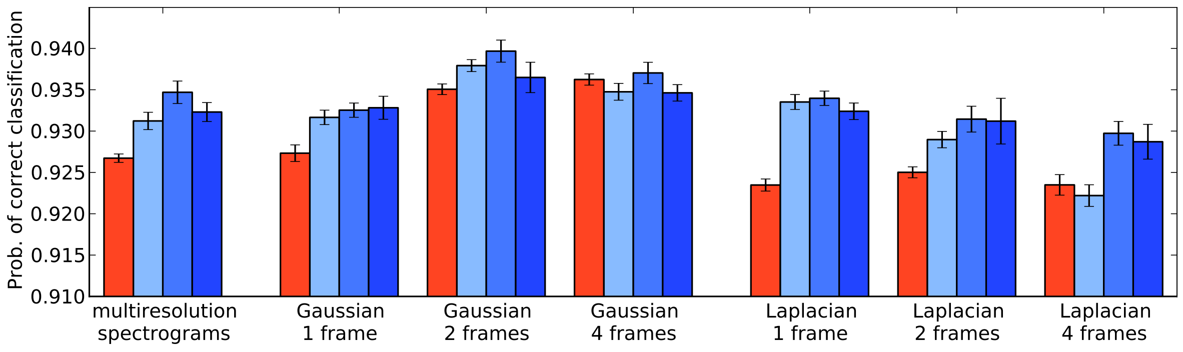


Figure 5: Results for the similarity metric learning task, for 28 different multiscale feature learning architectures. All reported results are averaged across 10 training runs with different initializations. Error bars indicate the standard deviation across these 10 runs.

scale representation. This does seem to help when using a Gaussian pyramid, however.

Results for the similarity metric task are shown in Figure 5. Here, a Gaussian pyramid with windows of 2 consecutive frames works best. Using 1000 means is noticeably worse than using 500 means. This can be attributed to the fact that the NCA objective seems to be very prone to overfitting when using a large amount of input features (6000 in this case, for 6 levels), despite our use of weight decay and early stopping for regularization.

5.2 Relevant timescales

To assess which timescales are relevant for different tags, we took the best architecture from the previous experiment and tried to predict tags from each level individually. Although a combination of all levels performs best for all tags, it is not always obvious precisely which timescales are the most relevant ones for a given tag. Figure 6 shows a selection of tags where some patterns can be identified.

Two tags describe the tempo of the music: *slow* and *fast*. As expected, the highest level seems to be the most important one for *slow*. For *fast*, level 1 (corresponding to a frame size of 128 ms) performs best. Both tags benefit quite a lot from the multiscale representation: a combination of all levels performs much better than any level individually. Tags describing dynamics, such as *loud*, *quiet* and *soft*, seem to rely mostly on the top level, corresponding to the coarsest timescale. This may also be because the top level is the only level in the Laplacian pyramid that is not a difference of two levels in the Gaussian pyramid.

Tags related to vocals can be predicted most accurately from intermediate levels, as evidenced by the results for *vocal*, *female*, *singing* and *vocals*. Of these, *female* is the easiest to predict, being the most specific tag. Finally, the *flute* tag is somewhat atypical among the tags describing instruments, in that it is the only one that relies mostly on the coarsest timescale (results for other instruments are not shown). A possible reason for this could be that the instrument lends itself well to playing longer, drawn out notes. A quick examination of the dataset reveals that many examples tagged *flute* in the dataset feature such notes.

6. CONCLUSION AND FUTURE WORK

We have proposed three approaches to building multiscale feature learning architectures for music audio, and we have evaluated them using two tasks designed to demonstrate the versatility of the learned features. Although learning features with the spherical K-means algorithm consistently improves results over just using PCA components, there is no clear winner among the proposed multiscale architectures. However, it is clear that learning features at multiple timescales improves performance over single-timescale approaches. We have also shown that different kinds of tags tend to rely on different timescales. Finally, we have

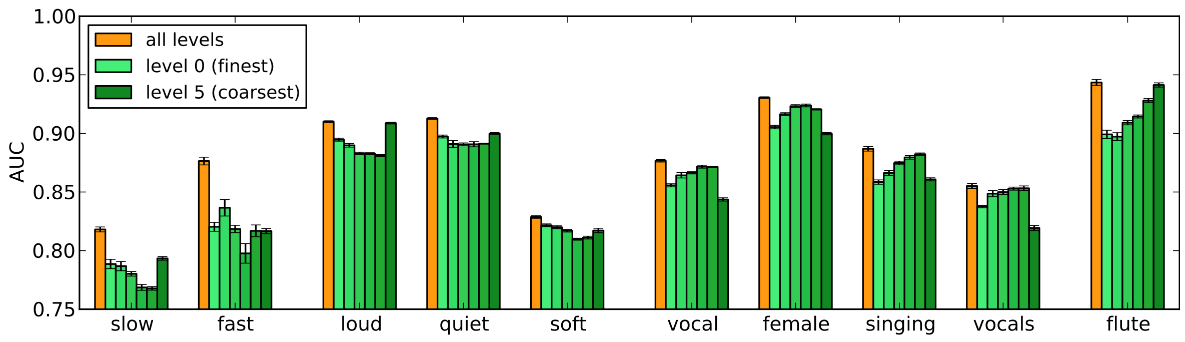


Figure 6: Results for a selection of individual tags, when using features from all levels combined and when using features from each level individually. The reported AUCs are for K-means with 500 means on the Laplacian pyramid with features learned from single frames.

observed that special care has to be taken to prevent overfitting when using large feature representations, which is almost unavoidable if a multiscale representation is desired.

In future work, we would like to improve the feature learning pipeline, by learning multiple layers of features for each timescale and investigating other encoding schemes. We would also like to evaluate the proposed architectures on an extended range of tasks, including content-based music recommendation and artist recognition, and on multiple datasets.

7. REFERENCES

[1] Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[2] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[3] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011.

[4] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means. *Neural Networks: Tricks of the Trade, Reloaded*, 2012.

[5] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.

[6] G. E. Dahl, Dong Yu, Li Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, January 2012.

[7] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[8] Rémi Foucard, Slim Essid, Mathieu Lagrange, Gaël Richard, et al. Multi-scale temporal fusion by boosting for music classification. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[9] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520, 2004.

[10] Philippe Hamel, Yoshua Bengio, and Douglas Eck. Building musically-relevant audio features through multiple timescale representations. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, 2012.

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

[12] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[13] Eric J. Humphrey, Juan P. Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, 2012.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012.

[15] Edith Law and Luis von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the 27th international conference on Human factors in computing systems*, 2009.

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

[17] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

[18] Malcolm Slaney, Kilian Q. Weinberger, and William White. Learning a metric for music similarity. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[19] Yichuan Tang and Abdel rahman Mohamed. Multiresolution deep belief networks. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.