TEMPORAL POOLING AND MULTISCALE LEARNING FOR AUTOMATIC ANNOTATION AND RANKING OF MUSIC AUDIO

Philippe Hamel, Simon Lemieux, Yoshua Bengio
DIRO, Université de Montréal
Montréal, Québec, Canada
{hamelphi,lemiesim,bengioy}@iro.umontreal.ca

Douglas Eck
Google Inc.
Mountain View, CA, USA
deck@google.com

ABSTRACT

This paper analyzes some of the challenges in performing automatic annotation and ranking of music audio, and proposes a few improvements. First, we motivate the use of principal component analysis on the mel-scaled spectrum. Secondly, we present an analysis of the impact of the selection of pooling functions for summarization of the features over time. We show that combining several pooling functions improves the performance of the system. Finally, we introduce the idea of multiscale learning. By incorporating these ideas in our model, we obtained state-of-the-art performance on the Magnatagatune dataset.

1. INTRODUCTION

In this paper, we consider the tasks of automatic annotation and ranking of music audio. Automatic annotation consists of assigning relevant word descriptors, or tags, to a given music audio clip. Ranking, on the other hand, consists of finding an audio clip that best corresponds to a given tag, or set of tags. These descriptors are able to represent a wide range of semantic concepts such as genre, mood, instrumentation, etc. Thus, a set of tags provides a high-level description of an audio clip. This information is useful for tasks like music recommendation, playlist generation and measuring music similarity.

In order to solve automatic annotation and ranking, we need to build a system that can extract relevant features from music audio and infer abstract concepts from these features. Many content-based music recommendation systems follow the same recipe with minor variations (see [5] for a review). First, some features are extracted from the audio. Then, these features are summarized over time. Finally, a classification model is trained over the summarized features to obtain tag affinities. We describe several previous approaches that follow these steps and have been applied to the Magnatune dataset [13] in Section 3.1. We then present an approach that deviates somewhat from the standard recipe by integrating learning steps before and after the temporal summarization.

This paper has three main contributions. First, we describe a simple adaptive preprocessing procedure of the music audio that incorporates only little prior knowledge on the nature of music audio. We show that the features obtained through this adaptive preprocessing give competitive results when using a relatively simple classifier. Secondly, we study the impact of the selection and mixing of pooling functions for summarization of the features over time. We introduce the idea of using min-pooling in conjunction with other functions. We show that combining several pooling functions improves the performance of the system. Finally, we incorporate the idea of multiscale learning. In order to do this, we integrate feature learning, time summarization and classification in one deep learning step. Using this method, we obtain state-of-the-art performance on the Magnatagatune dataset.

The paper is divided as follows. First, we motivate our experiments in Section 2. Then, we expose our experimental setup in Section 3. We present and discuss our results in Section 4. Finally, we conclude in Section 5.

2. MOTIVATION

2.1 Choosing the right features

Choosing the right features is crucial for music classification. Many automatic annotation systems use features such as MFCCs [8,12] because they have shown their worth in the speech recognition domain. However, music audio is very different from speech audio in many ways. So, MFCCs, which have been engineered for speech analysis might not be the optimal feature to use for music audio analysis.

Alternatives have been proposed to replace MFCCs. Recent work have shown that better classification performance can be achieved by using mel-scaled energy bands of the

spectrum [4]. Octave-based spectral contrast features [11] have been shown to also outperform MFCCs for genre classification. Thus, finding optimal features for audio classification is still an open problem.

In section 3.3, we present a relatively simple audio preprocessing based on spectral energy bands and principal component analysis (PCA).

2.2 Summarization of the features over time

Another important aspect of any automatic tagging system working on music audio is the question of the summarization of features over time, potentially allowing one to map a variable-length sequence into a fixed-size vector of features that can be fed to a classifier. The objective of summarization is to transform a joint feature representation into a more useful one that preserves important information while discarding noise, redundancy or irrelevant information. Summarizing features either in space (e.g. in visual recognition), or in time (e.g. in audio analysis) yields representations that are compact, invariant to shifts in space or time and robust to clutter.

One of the most straightforward ways to summarize features is feature pooling. Pooling consists in extracting simple statistics such as the mean or maximum of the features over an excerpt of a given time length. The choice of the pooling function has a great impact on the performance of the system. In [7], feature pooling in the domain of visual recognition is analyzed. The authors come to the conclusion that, depending on the data and features, neither max-pooling or mean-pooling might be optimal, but something in between might be. This underlines the importance of a thorough analysis of pooling functions for the specific task of music audio classification.

The choice of the temporal scale at which the pooling is applied also has a great impact on a system's performance. If we choose a time-scale that is too long, we discard too much information in the process, and the performance of the system suffers. If we choose a time-scale that is too small, the representation becomes less compact and looses the temporal shift invariance. It is possible to use onset detection to determine an optimized aggregation window length [20]. However, this method relies on onset detection methods which are not always reliable in all types of music.

2.3 Feature Learning and Deep Learning

It has been argued that features extracted by task-specific signal processing might be replaced by features learned over simpler low-level features, i.e., for object recognition [2, 15]. For instance, features learned with a Deep Belief Network over spectral amplitudes has been shown to outperform MFCCs for genre recognition and automatic annotation [10, 16].

Feature learning consists in exploiting the structure of the data distribution to construct a new representation of the input. This representation can be considered as a set of latent variables within a probabilistic model of the input. The transformation can be learned via unsupervised or supervised learning. Feature learning allows one to build systems relying less on prior knowledge and more on data, which grants more flexibility to adapt to a given task.

Deep learning algorithms attempt to discover multiple levels of features or multiple levels of representation. Several theoretical results and arguments [1] suggest that shallow architectures (with 1 or 2 levels, as in SVMs with a fixed kernel, for example) may be less efficient at representing functions that can otherwise be represented compactly by a deep architecture. The advantage of a deep architecture is that concepts or features at one level can be represented by combining features at lower levels, and these low-level features can be re-used in exponentially many ways as one considers deep architectures.

Convolutional Neural Networks (CNN) [15] were the first deep models to be applied successfully to real-world problems such as character recognition. CNNs present a hierarchical structure. Inserting a feature pooling layer between convolutional layers allows different layers of the network to work at different time scales and introduces more and more translation invariance (as well as robustness to other kinds of local distortions) as one moves up the hierarchy of the architecture. Hierarchical network structures such as CNNs seem ideal for representing music audio, since music also tends to present this hierarchical structure in time and different features of the music may be more salient at different time scales. Thus, in Section 3.5.2, we propose a hierachical model strongly inspired by CNNs.

3. EXPERIMENTAL SETUP

3.1 Magnatagatune Dataset

The Magnatagatune dataset consists of 29-second clips with annotations that were collected using an online game called TagATune. This dataset was used in the MIREX 2009 contest on audio tag classification [14]. In our experiments, we used the same set of tags and the same train/test split as in the contest. The training, valid and test set were composed of 14660, 1629 and 6499 clips respectively. The clips were annotated with a set of 160 tags, each clip being associated with between 1 and 30 tags.

We describe here the systems used by the four best contestants: Marsyas [19], Mandel [17], Manzagol [18] and Zhi [9]. All submissions use MFCCs as features, except for Mandel, which instead uses a cepstral transform that is closely related to MFCCs. Mandel also computes a set of temporal features. In addition, Marsyas includes a set of spectral features: spectral centroid, rolloff and flux. Zhi

uses Gaussian Mixture Models to obtain a song-level representation and uses a semantic multiclass labeling model. Manzagol summarizes the features with vector quantization (VQ) and applies an algorithm called PAMIR (passive-aggressive model for image retrieval). Mandel trains balanced SVMs for each tag. Finally, Marsyas uses running means and standard deviations of the features as input to a two-stage SVM classifier.

3.2 Performance evaluation

To evaluate the performance of our model, we compute the Area Under the ROC Curve (AUC). The ROC curve of a classifier is defined by the ratio of true positives over the positive outputs in function of the ratio of false positives over the negative outputs. The AUC gives the probability that, given one random positive and one random negative example, the classifier will rank the positive one higher than the negative one. Since the AUC is defined for a binary classification, and our task requires multi-label classification, there are two ways we can compute the AUC. By computing the average of the AUC for each tag (AUC-tag), we obtain a global measure of how good a classifier is at ranking clips given a tag (e.g. Which clip is more 'Reggae'?). Alternatively, we can compute the average of the AUC for each clip (AUC-clip) to obtain a measure of how good classifier is at ranking tags for a given clip (e.g. Is this clip more 'sad' or 'metal' ?).

Another measure which is closely related to the AUC is the precision at k where k is an integer. Given an ordered list of tags for a clip, it is defined by the ratio of true positives in the top k positions.

3.3 Audio Preprocessing

Our audio preprocessing involves three steps: discrete Fourier transform (DFT), mel-compression and principal component analysis whitening (PCA).

Firstly, to transform the audio in the spectral domain, we compute DFTs over windows of 1024 samples on audio at 22.1 KHz (i.e. roughly 46ms) with a frame step of 512 samples. Then, we run the spectral amplitudes through a set of 128 mel-scaled triangular filters to obtain a set of spectral energy bands. We compute the principal components of a random sub-sample of the training set and throw away only the components with very low variance (low eigenvalues), yielding 120 components in total. In order to obtain features with unitary variance, we multiply each component by the inverse square root of its eigenvalue, a transformation known as PCA whitening. We will refer to the preprocessed audio features as Principal Mel-Spectrum Components (PMSC).

3.4 Pooling functions

In our experiments, we used a set of pooling functions and some of their combinations. The functions we used are: mean, variance (var), maximum (max), minimum (min), and 3rd and 4th centered moments. The ith centered moment is defined by: $\frac{1}{n}\sum_{i=1}^{n}(x-\bar{x})^i$. By this definition, the variance corresponds to the second centered moment.

3.5 Models

We used two different models in our experiments. The first one, described in Section 3.5.1, is a rather conventional system that applies feature extraction, pooling and classification in three separate steps. The second one, described in Section 3.5.2, applies learning both before and after the temporal pooling. The models are illustrated in Figure 1.

3.5.1 Pooled Features Classifier (PFC)

The first model we evaluate applies a given set of pooling functions to the PMSC features, and sends the pooled features to a classifier. Each pooling window is considered as a training example for the classifier, and we average the predictions of the classifier over all the windows of a given clip to obtain the final classification. The classifier is a single hidden layer neural network, also known as multi-layer perceptron (MLP). We used a hidden layer of 1000 units, sigmoid activation, L2 weight decay and cross-entropy cost. We chose to use the MLP as a classifier for three main reasons. First, the hidden layer of the MLP should allow the model to learn dependencies between tags. Second, the MLP training time scales well (sub-linearly) with the size of the training set. Third, neural networks such as the MLP allows great flexibility in the structure of the network. This will allow us to extend the model to a multiscale structure, as we will see in section 3.5.2. We will refer to this model as the Pooled Features Classifier (PFC) model.

3.5.2 Multi-Time-Scale Learning model (MTSL)

The second model is structurally similar to the first one, except for the fact that we add a hidden layer between the input features and the pooling function. Thus the pooling is now applied on the activation of this new hidden layer. In this manner, the model is able to learn a representation of the features to be pooled. The weights connecting the input to the first layer are shared across all frames. We keep the same MLP structure as in the PFC model on top of the pooling. As for the PFC model, learning is purely supervised. During training, the error is back-propagated from the MLP, through the pooling functions, down to the first hidden layer. Thus, it is required to choose pooling functions for which a gradient can be defined, which is the case for all the functions described in section 3.4.

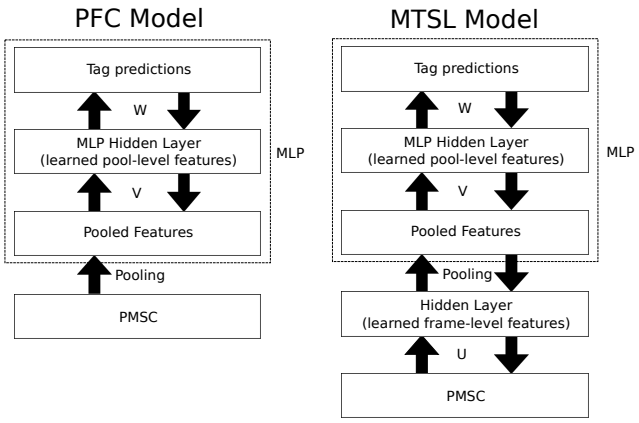

Figure 1. Comparison of the PFC and the MTSL model. Upward arrows represent the flow of feed-forward information. Downward arrows illustrate the flow of the error back-propagation. U, V and W are weight matrices to be learned.

In this model, while the first layer is learning on frames at a time scale of about 46ms, the second layer works at the scale of the pooling window. Since this model learns on different time scales, we will refer to it as the Multi-Time-Scale Learning (MTSL) model.

4. RESULTS AND DISCUSSION

We ran a few experiments to understand how much each piece of the puzzle contributes to the performance of the system. First, we evaluated how much the PCA step in the preprocessing improves the input representation. Then, we tested the performance of the system vs. the length of the pooling window. Afterwards, we compared different pooling functions and combined them for maximum performance. Finally, by adding a hidden layer to our model before the pooling, we trained a multiscale learning model.

In most experiments, we present the AUC-tag as our performance measure. Since it was the most stable valid measure during training, we chose it as our early-stopping criterion. However, the AUC-clip and precision at k tend to follow the same trend as the AUC-tag (i.e. good ranking models also give good annotations).

4.1 PCA

We measure the effect of the PCA on the mel-spectrum. We applied the PFC model on the features with and without PCA as well as MFCCs for comparison. Results are shown in Table 1. We can see that the mel-spectrum features perform better than MFCCs, and that adding the PCA step further improves performance, as well as greatly reducing training time.

It has been shown in [4] that using the full covariance matrix of spectral energy bands improves classification performance. The PCA whitening uncorrelates the spectral features, and thus encapsulate most information in the diagonal of the covariance matrix. In consequence, relevant information flows better through the pooling functions, which gives better pooled features and allows faster and more efficient training.

	valid AUC-tag	mean time
MFCC(20)	0.77 +/- 0.04	5.9h
Mel-spectrum(128)	0.853 +/- 0.008	5.2h
PMSC(120)	0.876 +/- 0.004	1.5h

Table 1. Mean performance (higher is better) and mean training time of different features on the PFC model. In parantheses is indicated the dimensionality of the input

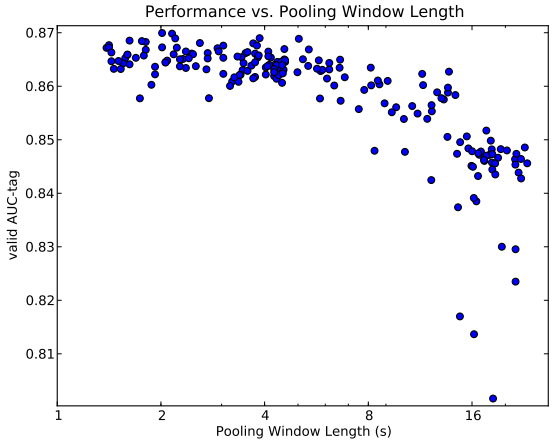

Figure 2. Performance w.r.t. length of pooling window.

4.2 Finding the optimal pooling window

In order to find the best pooling time scale for our task, we trained a set of PFC models using different pooling windows. The results on the validation set is shown in Figure 2. We see that the performance reaches a plateau when the pooling window is around 2.3 seconds. The models illustrated in the figure used a combination of mean, variance and maximum pooling, but the same tendency was obtained with other pooling functions and combinations.

4.3 Pooling functions

We compared the performance of different pooling functions and some of their combinations on the PFC model. For each type of pooling we trained 10 models with the same distribution of hyper-parameters. The results are illustrated in Figure 3. The label `all moments` refer to the combination of mean, variance and 3rd and 4th centered moments. We see that the max and min functions perform well by themselves. The third and fourth centered moments give poor results. Even when combined with other pooling functions, they hinder performance. Combining mean, variance,

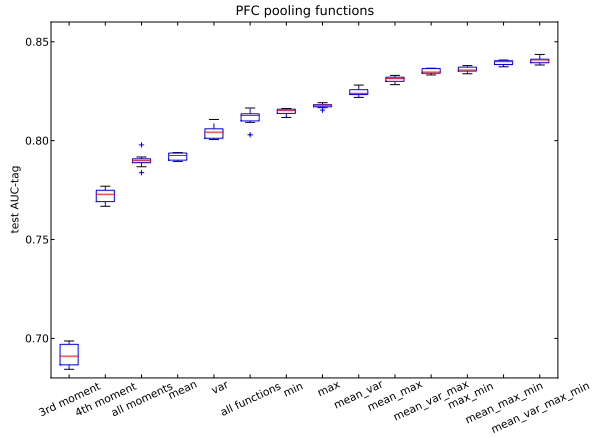

Figure 3. Performance of different combinations of pooling functions for the PFC model

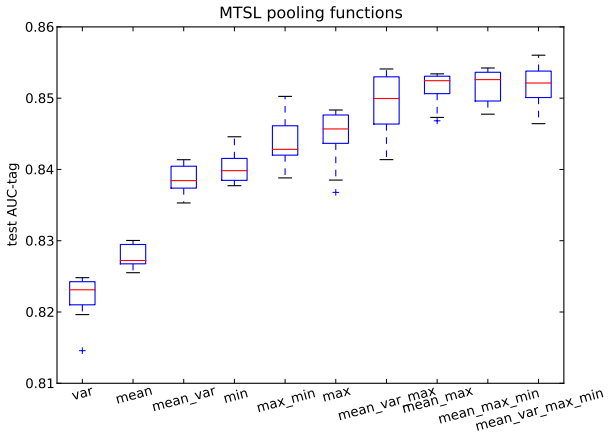

Figure 4. Performance of different combinations of pooling functions for the MTSL model

maximum and minimum gave the best performance.

4.4 Multiscale learning

We trained sets of MTSL models with different pooling functions combinations. For this experiment, we fixed the pooling window at about 2.3 seconds, following the results from Section 4.2. The results for different sets of pooling functions is given in Figure 4. We see that, once again, combining pooling functions gives better classification performance. In particular, all the models that combined mean and max pooling tend to perform better than others. Also, variance pooling seems to perform worse than other pooling functions. It helps when combined with the mean, but it does not give any significant improvement when combined with max and min pooling.

One might think that models combining pooling functions would require more time to train. However, there was no significant difference in training time for the different pooling combinations, except for `var` and `mean_var` that required more time. This can be explained by the fact that, even though the number of pooled features is greater, the combination of pooling functions allows the error information to flow better to the first layer, thus facilitating learning.

We used between 100 and 200 units in the first layer for the experiments presented in Figure 4. Using more units further improves performance, but requires more computing time. The best MTSL models used around 350 units.

4.5 Comparative test performance

We compare the results of our models to those of the MIREX 2009 contest [1]. In Table 2, we report the test performance of models that performed best on the validation dataset. We see that, even without multiscale learning, PMSC features with the PFC model outperform the best results from the competition. Applying multiscale learning gives an additional boost to the performance.

5. CONCLUSION

In this paper we have proposed a few improvements for automatic annotation and ranking systems:

- We introduced the PMSC features and demonstrated their performance.
- We demonstrated how combining pooling functions helps learning.
- We proposed the MTSL model, adding multiscale structure in a deep architecture, and it obtains state-of-the-art performance.

We have demonstrated step-by-step the positive impact of each of these elements. These conclusions were demonstrated on the task of automatic music annotation and ranking, but may be transferable to other MIR task.

The MTSL model we proposed presents a relatively simple hierarchical structure. There are many ways that we could still improve it further. For instance, using a deeper model with more time scales and smaller pooling windows might allow to learn a better representation of the music audio. Also, applying unsupervised training would probably improve the performance, especially for deeper models. Furthermore, the use of larger convolutional filters instead of our frame-by-frame hidden-layer could allow a richer representation of time dynamics. Another possible improvement would be to also use the time derivatives of the latent features as features to be pooled.

It would also be interesting to apply our model to a larger dataset such as the Million Song Dataset [6] to test how well it scales to much larger music databases.

[1] http://www.music-ir.org/mirex/wiki/2009:Audio_Tag_Classification_Tagatune_Results

Measure	Manzagol	Zhi	Mandel	Marsyas	Mel-spec+PFC	PMSC+PFC	PSMC+MTSL
Average AUC-Tag	0.750	0.673	0.821	0.831	0.820	0.845	**0.861**
Average AUC-Clip	0.810	0.748	0.886	0.933	0.930	0.938	**0.943**
Precision at 3	0.255	0.224	0.323	0.440	0.430	0.449	**0.467**
Precision at 6	0.194	0.192	0.245	0.314	0.305	0.320	**0.327**
Precision at 9	0.159	0.168	0.197	0.244	0.240	0.249	**0.255**
Precision at 12	0.136	0.146	0.167	0.201	0.198	0.205	**0.211**
Precision at 15	0.119	0.127	0.145	0.172	0.170	0.175	**0.181**

Table 2. Performance of different models for the TagATune audio classification task. On the left are the results from the MIREX 2009 contest. On the right are our results.

6. ACKNOWLEDGMENTS

The authors were financially supported by FQRNT and NSERC grants. The machine learning models presented in this paper were built using the Theano python library [3]. The authors would like to thank the Theano developper team.

7. REFERENCES

[1] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[2] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral.

[4] James Bergstra, Michael Mandel, and Douglas Eck. Scalable genre and tag prediction with spectral covariance. In *ISMIR*, 2010.

[5] T. Bertin-Mahieux, D. Eck, and M. Mandel. Automatic tagging of audio: The state-of-the-art. In *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010.

[6] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, 2011. (submitted).

[7] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *Intl. Conf. on machine learning (ICML)*, 2010.

[8] Shi-Huang Chen and Shih-Hao Chen. Content-based music genre classification using timbral feature vectors and support vector machine. In *Proc. Intl. Conf. on Interaction Sciences: Information Technology, Culture and Human*, ICIS. ACM, 2009.

[9] Zhi-Sheng Chen and Jyh-Shing Roger Jang. On the Use of Anti-Word Models for Audio Music Annotation and Retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(8), 2009.

[10] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *ISMIR*, 2010.

[11] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. *Music type classification by spectral contrast feature*. IEEE, 2002.

[12] Thibault Langlois. A music classification method based on timbral features. In *ISMIR*, 2009.

[13] Edith Law and Luis von 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.

[14] Edith Law, Kris West, Michael Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games: the case of music tagging. In *ISMIR*, 2009.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1:541–551, December 1989.

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

[17] M. I. Mandel and D. P. W. Ellis. Multiple-instance learning for music information retrieval. In *ISMIR*, 2008.

[18] Pierre-Antoine Manzagol and Samy Bengio. Mirex special tagatune evaluation submission. In *MIREX*, 2009.

[19] George Tzanetakis. Marsyas submissions to MIREX 2009. In *MIREX*, 2009.

[20] Kris West and Stephen Cox. Finding an optimal segmentation for audio genre classification. In *ISMIR*, 2005.