AUTOMATIC IDENTIFICATION OF INSTRUMENT CLASSES IN POLYPHONIC AND POLY-INSTRUMENT AUDIO

Philippe Hamel, Sean Wood and Douglas Eck
Département d'informatique et de recherche opérationnelle, Université de Montréal
CIRMMT
{hamelphi,woodsean,eckdoug}@iro.umontreal.ca

ABSTRACT

We present and compare several models for automatic identification of instrument classes in polyphonic and poly-instrument audio. The goal is to be able to identify which categories of instrument (Strings, Woodwind, Guitar, Piano, etc.) are present in a given audio example. We use a machine learning approach to solve this task. We constructed a system to generate a large database of musically relevant poly-instrument audio. Our database is generated from hundreds of instruments classified in 7 categories. Musical audio examples are generated by mixing multi-track MIDI files with thousands of instrument combinations. We compare three different classifiers : a Support Vector Machine (SVM), a Multilayer Perceptron (MLP) and a Deep Belief Network (DBN). We show that the DBN tends to outperform both the SVM and the MLP in most cases.

1. INTRODUCTION

Thanks in part to the vast amount of music available online, much research has been done on the automatic extraction of descriptors for music audio, such as genre, artist, mood and instrumentation. Because the majority of this research has focused on commercial recorded music, where ground truth is lacking, relatively little work has been done in identifying which instruments are playing in music audio. Solving this problem would give rise to better description of commercial audio collections. It could also form a part of a system able to synthesize music with timbres that match the instruments found in a particular audio file (e.g. "generate music that sound like this Sex Pistols mp3"). Such a system may be useful in applications such as user-content-guided video game music generation.

In this paper, our focus is on constructing a model able to determine which classes of musical instrument are present in a given musical audio example, without access to any information other than the audio itself. In order to obtain sufficient labeled training examples for good generalization, we generated our own database of audio. Our goal was to have enough variability in the set of instruments so as to allow us to generalize to instruments not used in the training set. An overview of our system is illustrated in Figure 1.

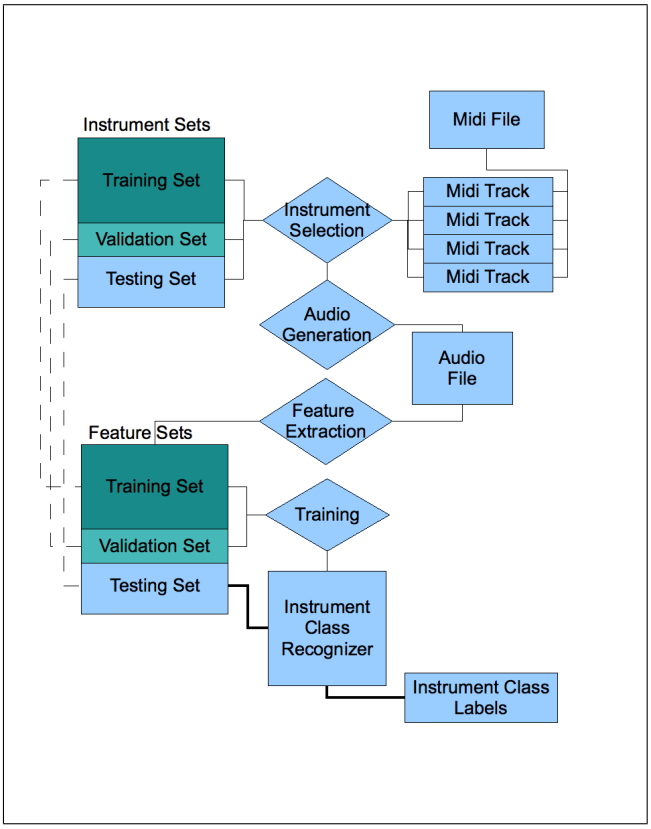

Figure 1. Overview of our automatic instrument class recognizer model

We compared three different classifiers to solve this task : a Support Vector Machine (SVM), a Multilayer Perceptron (MLP) and a Deep Belief Network (DBN).

The main contribution of this paper is the introduction of the DBN model to the instrument recognition task. Until recently, deep neural networks (i.e. networks having many hidden layers) were not used in practice because they are hard to train using random initialization and gradient descent alone. Recent developments have made training such networks possible [3, 15]. DBNs have since shown potential in many fields such as image and speech recognition.

Deep networks aim at learning higher-level features at each layer from the features of the layer below. Learning such high-level features allows a model to construct an abstract representation of the inputs. This is similar to how the human brain transforms raw sensory inputs to abstract features. An in-depth description and justification of the use of deep architectures for learning can be found in [2].

The paper is organized as follows : In Section 2, we describe previous research in the domain of instrument recognition. In Section 3 we describe the system used to generate our audio database. In Section 4, we discuss the features extracted from the audio. In Section 5, we describe in detail the three classification models we employed. We then discuss our results in Section 6.

2. PREVIOUS WORK

The problem of automatic instrument classification has been tackled from several different angles in the past decades. Psycho-acoustic studies have been conducted to build "timbre spaces" in which the distance between two sounds represents their degree of similarity [24, 30]. From the topology of these spaces, we can outline some important features of the sounds that are important when studying timbre (e.g. spectral centroid, spectral flux, etc.). A lot of work on instrument recognition has been done on isolated instrument sounds and monophonic audio [1,8–11,17,22,23,27]. An overview of previous approaches to automatic instrument classification is described in [13]. Recent work deals with more complex and musically relevant sounds such as duets and polyphonies [6,7,16,18–20]. There has also been much work done on the related task of predicting genre- and instrument-related tags from audio [4, 21, 26] for music recommentation.

In a polyphonic context, notes are not easily separable. Pitch tracking and source separation techniques can be useful to address this problem, but these are still unsolved problems in polyphonic audio that attracts a lot of research activity. The problem of instrument recognition becomes even more complex when we consider multi-instrument audio. Many different machine learning models have been tried to solve this task. In [6], missing feature theory and a Gaussian-mixture model (GMM) classifier was used to identify instruments in monophonic and polyphonic audio. In [20], a process using linear discriminant analysis (LDA) for instrument recognition for solo and duet performances is presented. A Support Vector Machine (SVM) and a hierarchical classification scheme are used on polyphonic music in [7]. [16] presents a classification model using LDA and feature weighting using polyphonic audio and musical context information. In [18], instrument recognition in polyphonic audio is done by applying a post-process on a mid-level harmonic atoms representation. [19] compares the performance of three classifiers : SVMs, Extra Trees and K-Nearest Neighbors.

Unfortunately, the relative performances of these different approaches are difficult to measure. Since each research team uses a different test database and a different classification taxonomy, it would be unfair to compare the reported classification accuracies.We take a small step towards addressing this by publishing our entire instrument database. See [12].

Most previous work on instrument recognition in polyphonic audio has focused on recognizing specific instruments from a small set of instruments. These models do not attempt to deal with instruments they have never heard before. In this paper we address this limitation by introducing a model capable of recognizing *classes* of instruments instead of specific ones. We argue that this behavior is better suited to large, rapidly-evolving commercial audio databases.

3. DATABASE GENERATION

To solve a difficult task such as instrument class recognition in poly-instrument audio, we require a large database with a lot of variability in the data. To address this challenge, we constructed our own database with a wide range of sampled and synthesized instruments. Using MIDI files to control our instruments, we were able to easily generate musically plausible examples with a wide range of velocities, harmonization, and note lengths. We believe this will help our classifier to better generalize.

3.1 Instrument bank

We used the instrument sounds from the commercial sampler "Kontakt 3" from "Native Instruments" to generate our database. The advantage of using a sampler instead of banks of isolated sounds or recorded performances is that we can generate musically relevant audio files from any MIDI file. We selected 172 different physical instruments. For 23 of these physical instruments, we treated different dynamics as individual instruments for a total of 320 instruments. We separated our instruments into 7 classes : Piano, Guitar, Bass, Organ, Woodwind, Brass and Strings. Each class contained from 9 to 94 instruments. To test for generalization we divided our instrument bank into three independent sets : 50% of the instruments were placed in a training set, 20% in a validation set and the remaining 30% in a test set.

3.2 Audio Generation

We built a system to automatically generate a large quantity of audio files using MIDI files and a bank of instrument samples. We generated two audio corpuses using solo instrument and poly-instrument MIDI files. We composed and pre-mixed six to seven 30-second midi files per experiment. The first corpus was obtained by generating audio from the solo instrument MIDI files, while the second was generated with multi-track MIDI files. We separated the MIDI files into individual tracks, each file having between two and six tracks. For each track, we randomly selected an instrument with a compatible range. By "compatible range", we mean that the chosen instrument has a sample set with a wide enough range (e.g. C4–C6) to actually play all of the notes in the file. We then mixed the tracks, making sure that all instruments in a mix were from the same

data set (train, valid or test). We generated audio from each MIDI file with hundreds of different instruments mixes.

Examples of our generated audio and corresponding model predictions are available at the website : `http://www.iro.umontreal.ca/~gamme/ismir_2009/`

4. FEATURE EXTRACTION

The selection of features is a crucial aspect of any instrument classifier. One of the most widely used features for timbre analysis is the Mel-Frequency Cepstral Coefficients (MFCCs). We used the 20 first MFCCs as well as their first and second derivatives (dMFCCs and ddMFCCs). The MFCCs were calculated on 32 ms windows with a window step size of 10 ms.

We also used a set of spectral features : centroid, spread, skewness, kurtosis, decrease, slope, flux and roll-off. The mathematical definitions of these features are described in [25].

We divided the audio files into 1 second frames, and calculated the mean and the standard deviation of each feature for each frame, yielding two values for each feature. In total, the feature vectors contain 136 values : 40 MFCCS, 40 dMFCCs, 40 ddMFCCs and 16 spectral features.

5. MODELS

We tested three different classifiers : a Mulitlayer Perceptron (MLP), a Support Vector Machine (SVM) and a Deep Belief Network (DBN).

5.1 Multilayer Perceptron

The first model is a single hidden layer feed-forward neural network, also know as Multilayer Perceptron. An advantage of such models is that they are very fast to use, once trained, making them good candidates for a real-time application. We used the neural network implementation from the publicly available PLearn library [28].We used a $tanh$ activation function for the hidden layer, and a logistic sigmoid function for the output layer. We used cross-entropy as the cost function to optimize. To avoid overfitting, we used an L2 norm regularization on the weights as well as an early stopping condition. We also used conjugate gradient descent to accelerate training.

5.2 Support Vector Machine

We also tested a Support Vector Machine (SVM) with a radial basis kernel. SVMs are widely used large margin classifiers. The implementation of a SVM is quite complex, but publicly available ready-to-use libraries make them rather simple to use [5]. SVMs have been used for the task of instrument recognition with a good degree of success [19]. SVMs have the advantage of having fewer hyper-parameters to optimize than neural networks. We used cross-validation to optimize the hyper-parameters.

5.3 Deep Belief Network

A deep network is constructed by superposing many layers of neurons. It is essentially an MLP with many hidden layers. The main difference comes from the initialization of the weights of the connections between neurons. In the single hidden layer case, a random initialization is generally sufficient for the gradient descent to work. However, with random initialization on many hidden layers, the solutions obtained appear to correspond to poor solutions that perform worse than the solutions obtained for networks with 1 or 2 hidden layers [2, 3]. To circumvent this problem, the DBN learning procedure consists of a greedy layer-wise unsupervised pre-training phase, followed by a supervised gradient descent fine-tuning phase. The pre-training phase configures the network such that it may efficiently represent the input data. The pre-training phase is typically done with layers of Restricted Boltzmann Machines (RBMs) [15] or autoencoders [14, 29]. In this work, we used RBMs. RBMs are constituted of two layers of neurons : a visible layer and a hidden layer. Each neuron is connected to every neuron of the other layer, but have no connection with neurons of the same layer. The RBMs have a simple and fast learning algorithm that basically try to minimize the reconstruction error using an algorithm called contrastive divergence [15]. We can stack many RBMs on top of each other, where the visible layer of the top RBMs is the hidden unit of the RBM below, to obtain a DBN. The pre-training phase then consists of training each RBM sequentially, starting from the input layer up to the output layer. Once this is completed, the model is further "fine tuned" for a specific supervised learning task. This fine tuning is done using the same gradient descent learning asn an MLP : given a cost function to optimize, the gradient is propagated through the network, and weights are updated accordingly. As for our MLP model, we used a cross-entropy cost function. One problem with DBNs is the large number of hyper-parameters : number of layers, number of units per layer, pre-training learning rate, gradient descent learning rate, weight regularization constant, number of pre-training epochs. This makes the hyper-parameter search tedious.

6. EXPERIMENTS

6.1 Experimental Setup

As in [19], we used weak labels as targets for training, i.e. targets for every frame in a given song are the same. If a song contains a string instrument and a guitar, every frame of that song will be labelled as containing 'strings' and 'guitar', even though there is no guarantee that there is a string instrument and a guitar in every frame.

The task of instrument class recognition in poly-instrument audio is a multi-label classification task, i.e. each instrument class may be present or not, and the classifier is unaware of how many classes are present.

In order to compare the three different models, we used the F-Score as a performance measure. The F-Score is a measure that balances precision and recall. The precision

and recall are defined as

$$\text{Precision} = \frac{tp}{tp+fp}, \text{Recall} = \frac{tp}{tp+fn} \quad (1)$$

where tp, fp and fn are the number of 'true positives', 'false positives' and 'false negatives' examples. A true positive is a positive example that was correctly labeled as positive by the model. A false positive is a negative example that was mislabeled as positive. A false negative is a positive example that was mislabeled as negative. The F-Score (F) is defined as the harmonic mean of the precision and the recall

$$F = \frac{2(\text{precision} * \text{recall})}{\text{precision} + \text{recall}} \quad (2)$$

This can be simplified to

$$F = \frac{2tp}{2tp + fp + fn}. \quad (3)$$

To obtain F-Scores for each instrument, we calculated the F-Scores independently as for 7 independent classification tasks. In order to get a global F-score that represents the overall performance of the models, we took the sum of tp, fp and fn over all the instruments.

The neural networks (MLP, DBN) output a probability $\in [0, 1]$for each instrument, representing the network's belief that the instrument is present in the given input frame. If the probability for a given instrument is higher than a given threshold, we classify this class as being present. Lowering the threshold improves the recall, but lowers the precision, while increasing the threshold has the opposite effect. To label a whole song, we take the mean of the probabilities from each frame and apply a threshold to decide whether or not each instrument class is present. We optimized the threshold to maximize the global F-score.

The output of our SVM model is binary (0 or 1) for each class. We used a similar technique as the neural network to label a song, except that we have binary votes instead of probabilities.

6.2 Results and Discussion

6.2.1 Feature sets

To confirm that the features we extracted from the audio were useful for training our models, we compared the results of training with subsets of our feature sets on the solo instrument audio corpus. The mean F-score for each subset using our three models are shown in Table 1. We see a tendency that using more features helps the SVM and the DBN, but the MLP doesn't show improvement with the full set of features compared to using only 20 MFCCs. Another result that is remarkable is that the DBN performs surprisingly well compared to the two other models with only the spectral features as inputs. For the following experiments, we will always use our full set of features.

6.2.2 Solo instrument audio

Our first audio corpus contains solo performances from all the instruments. For this experiment, we generated a total of 2735 song examples generated from 7 different MIDI files. We used 1984 of these for training and validation, for a total of 62434 1-second frames. The results are shown in Table 2.

	SVM	MLP	DBN
Spectral Features (16)	0.51	0.74	0.81
12 MFCCs (72)	0.75	0.85	0.85
20 MFCCs (120)	0.81	0.86	0.87
All Features (136)	0.84	0.84	0.88

Table 1. Global F-score for different features subsets (features vector length in parenthesis)

	SVM	MLP	DBN	%
Bass	0.88	0.88	0.88	13.85%
Brass	0.87	0.88	0.91	22.37%
Guitar	0.0	0.0	**0.21**	2.13%
Organ	0.96	0.89	0.96	7.46%
Piano	0.45	0.43	**0.57**	6.39%
Strings	0.94	0.95	0.97	9.59%
Woodwind	0.82	0.85	0.89	29.83%
Global	0.84	0.84	0.88	

Table 2. F-score for solo instrument audio. The results that clearly outperforms the other models are highlighted in bold. The percentage of positive examples in the training set for each instrument is shown in the rightmost column

We see that the DBN tends to perform better than both the SVM and the MLP in this experiment. Moreover, the DBN seems to perform significantly better when the quantity of positive training example is smaller. Note that both the SVM and the MLP were unable to recognize the guitar instrument class. This is probably related to the fact that only a small fraction of the data set contained positive guitar examples.

The DBN that gave the best validation F-score had 5 layers of 50 units each. Only 3 epochs of pre-training over the training set were necessary to achieve the best generalization performance. The best MLP model had 40 hidden units.

6.2.3 Poly-instrument audio

Our second audio corpus is constructed from mixes of instruments. Each song is generated from one of 6 MIDI files containing between 2 and 6 tracks, and thus each example contains from 1 to 6 classes (many instruments from the same class are allowed). The data set is constituted of 3654 training and validation examples divided in 186532 frames. Results are shown in Table 3.

Again, in this experiment, the DBN seems to perform slightly better than the SVM and the MLP. In three cases (brass, guitar and woodwind), the performance difference was important. The DBN with the best generalization performance in this experiment had 4 layers of 100 units and

	SVM	MLP	DBN	%
Bass	0.86	0.83	0.85	50.00%
Brass	0.38	0.45	**0.63**	25.90%
Guitar	0.05	0.15	**0.28**	11.94%
Organ	0.84	0.84	0.85	62.99%
Piano	0.83	0.80	0.83	64.44%
Strings	0.37	0.37	0.36	18.82%
Woodwind	0.31	0.41	**0.52**	31.81%
Global	0.72	0.72	0.74	

Table 3. F-score for poly-instrument audio. The results that clearly outperforms the other models are highlighted in bold. The percentage of positive examples in the training set for each instrument is shown in the rightmost column

required 4 epochs of pre-training. The best MLP was constructed with 60 hidden units.

6.3 Discussion

In all 3 experiments, the DBN generally performed better than the 2 other models, although the difference is not always important. The DBN tends to perform better especially in cases where the quantity of positive examples is small. This could indicate that the DBN was able to learn higher-level features to discriminate instrument classes. In other words, it was able to use what it learned from other instrument classes to discriminate instruments that were less frequent.

Although the results seem to show that the DBN performed better than the SVM and MLP, we cannot draw any hard conclusion with these results because of the similarity of the results and the lack of confidence intervals. The F-Score may not be the best measure to get such confidence intervals. However, these results clearly show that DBNs can be useful for the task of instrument recognition. These results also motivate more experiments to confirm the tendency shown. In future work, these experiments should be run using cross-fold testing and measuring the classification error in order to obtain a reliable confidence measure.

When generating our labeled examples, we tried to stay as close to real music as possible. The MIDI format is good to reproduce some features of real music such as harmonization and timing. However, it is harder to represent musical features such as expressiveness and instrument dynamics variations in MIDI. Also, our system used a rather simple fixed mixing of the instruments in a given song, which gave rise to small variability in the relative volume of the instruments. The limited number of midi files we used is also a limitation of our model. In future work, we would like to add more variability to the music generation by using more songs and by diversifying the mixing between instruments.

Another aspect that could improve the performance of the three models would be to learn an independent decision threshold for each instrument class. We used only one decision threshold that was optimized on the validation set global F-Score. This may be related to the fact that the SVM and the MLP were unable to recognize the guitar class in the solo instrument experiment.

7. CONCLUSION AND FUTURE WORK

In this work, we have introduced the DBN model for instrument recognition. We have shown that DBNs perform at least as well as SVMs and MLPs for this task. We have also shown that the DBN tends to outperform these models when the feature set is limited, and when the number of positive examples for a class is limited. These results motivate the application of deep networks in music information retrieval tasks.

As seen in Section 4, adding more relevant features seems to improve the performance of the classifiers. In future work, it would be interesting to consider extracting a wider variety of features from the audio. In this study, we avoided harmonic features that rely on the identification of a single fundamental frequencey for a frame of audio because this is ill-defined in the polyphonic case. In future work, it would be interesting to test if extracting simple harmonic features (e.g. odd to even harmonics ratio) from mixed instruments using an estimate of the most salient frequency could help for this task. We suppose that there is useful information in such features.

We also plan to add more variability to our data set by adding reverb and background noise to our audio examples. We hypothesize that this would add robustness to our trained models.

Finally, it would be interesting to test our model on real music. This is something that we plan for the near future. To test our model on commercial music, we would need to train a wider range of instruments, such as drums, distorted guitars, vocals, etc.

8. ACKNOWLEDGMENTS

Philippe Hamel was supported financially by FQRNT. Douglas Eck and Sean Wood are financed by NSERC Discovery Grants. Special thanks to Nathanael Lecaude, Pascal Lamblin, Simon Lemieux, Olivier Delalleau, and all the people at the GAMME and LISA labs for helpful discussions.

9. REFERENCES

[1] G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. *EURASIP J. Appl. Signal Process.*, 2003 :5–14, 2003.

[2] Y Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, to appear, 2009.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[4] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger : A model for predicting social tags from

acoustic features on large music databases. *Journal of New Music Research"*, 37(2) :115–135, 2008.

[5] C.-C. Chang and C.-J. Lin. *LIBSVM : a library for support vector machines*, 2001.

[6] J. Eggink and G.J. Brown. Application of missing feature theory to the recognition of musical instruments in polyphonic audio. In *International Symposium on Music Information Retrieval (ISMIR '03)*, 2003.

[7] S. Essid, G. Richard, and B. David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions On Audio, Speech And Language Processing*, 14 :68–80, 2006.

[8] S. Essid, G. Richard, and B. David. Musical instrument recognition by pairwise classification strategies. *IEEE Transactions On Audio, Speech And Language Processing*, 14 :1401–1412, 2006.

[9] A. Fraser and I. Fujinaga. Toward realtime recognition of acoustic musical instruments. In *Proceedings of the International Computer Music Conference. 175–7.*, 1999.

[10] I. Fujinaga. Machine recognition of timbre using steady-state tone of acoustic musical instruments. In *Proceedings of the International Computer Music Conference. 207-10.*, 1998.

[11] I. Fujinaga and K. MacMillan. Realtime recognition of orchestral instruments. In *Proceedings of the International Computer Music Conference. 141–3.*, 2000.

[12] P. Hamel, S. Wood, S. Lemieux, and D. Eck. The GAMME Poly-Instrument Audio Database, 2009. `http://www.iro.umontreal.ca/~gamme/instrument_data/`.

[13] P. Herrera, A. Klapuri, and M. Davy. *Signal Processing Methods for Music Transcription*, chapter Automatic Classification of Pitched Musical Instrument Sounds, pages 163–200. Springer US, 2006.

[14] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313 :504–507, 2006.

[15] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

[16] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Instrument identification in polyphonic music : feature weighting to minimize influence of sound overlaps. *EURASIP J. Appl. Signal Process.*, 2007(1) :155–155, 2007.

[17] A. G. Krishna and T. V. Sreenivas. Music instrument recognition : from isolated notes to solo phrases. In *in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '04)*, volume 4, pages 265–268, Montreal, Quebec, Canada, May 2004.

[18] P. Leveau, D. Sodoyer, and Daudet L. Automatic instrument recognition in a polyphonic mixture using sparse representations. In *ISMIR*, 2007.

[19] D. Little and B. Pardo. Learning musical instruments from mixtures of audio with weak labels. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[20] A. Livshin and X. Rodet. Musical instrument identification in continuous recordings. In *Proc. of the 7th Int. Conference on Digital Audio Effects (DAFX-04), Naples, Italy*, 2004.

[21] M. I. Mandel and D. P.W. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 594–599, 2005.

[22] J. Marques and P. J. Moreno. A study of musical instrument classification using gaussian mixture models and support vec- tor machines. Crl technical report series crl/4, Cambridge Research Laboratory, Cambridge, Mass, USA, 1999.

[23] K Martin. *Sound-source recognition : A theory and computational model*. PhD thesis, MIT, 1999.

[24] S. McAdams. Psychological constraints on form-bearing dimensions in music. *Contemporary Music Review*, 1989.

[25] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2004.

[26] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech And Language Processing*, 16(2) :467–476, 2008.

[27] E. Vincent and X. Rodet. Instrument identification in solo and ensemble music using independent subspace analysis. In *ISMIR*, 2004.

[28] P. Vincent, Y. Bengio, N. Chapados, et al. Plearn. `http://plearn.berlios.de/`.

[29] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML '08 : Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, New York, NY, USA, 2008. ACM.

[30] D. L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, Vol 3 No 2, 1979.