Université de Montréal

Apprentissage de représentations musicales à l'aide d'architectures profondes et multiéchelles

par
Philippe Hamel

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

mai, 2012

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

Apprentissage de représentations musicales à l'aide d'architectures profondes et multiéchelles

présentée par:

Philippe Hamel

a été évaluée par un jury composé des personnes suivantes:

Pascal Vincent,	président-rapporteur
Douglas Eck,	directeur de recherche
Yoshua Bengio,	codirecteur
Sylvie Hamel,	membre du jury
Ichiro Fujinaga,	examinateur externe
Nathalie Fernando,	représentante du doyen de la FAS

Thèse acceptée le: . . . . . .

RÉSUMÉ

L'apprentissage machine (AM) est un outil important dans le domaine de la recherche d'information musicale (*Music Information Retrieval* ou MIR). De nombreuses tâches de MIR peuvent être résolues en entraînant un classifieur sur un ensemble de caractéristiques. Pour les tâches de MIR se basant sur l'audio musical, il est possible d'extraire de l'audio les caractéristiques pertinentes à l'aide de méthodes traitement de signal. Toutefois, certains aspects musicaux sont difficiles à extraire à l'aide de simples heuristiques. Afin d'obtenir des caractéristiques plus riches, il est possible d'utiliser l'AM pour apprendre une représentation musicale à partir de l'audio. Ces caractéristiques apprises permettent souvent d'améliorer la performance sur une tâche de MIR donnée.

Afin d'apprendre des représentations musicales intéressantes, il est important de considérer les aspects particuliers à l'audio musical dans la conception des modèles d'apprentissage. Vu la structure temporelle et spectrale de l'audio musical, les représentations profondes et multiéchelles sont particulièrement bien conçues pour représenter la musique. Cette thèse porte sur l'apprentissage de représentations de l'audio musical. Des modèles profonds et multiéchelles améliorant l'état de l'art pour des tâches telles que la reconnaissance d'instrument, la reconnaissance de genre et l'étiquetage automatique y sont présentés.

Mots clés : apprentissage machine, recherche d'information musicale, analyse d'audio musical, étiquetage automatique, apprentissage profond, apprentissage multiéchelle.

ABSTRACT

Machine learning (ML) is an important tool in the field of music information retrieval (MIR). Many MIR tasks can be solved by training a classifier over a set of features. For MIR tasks based on music audio, it is possible to extract features from the audio with signal processing techniques. However, some musical aspects are hard to extract with simple heuristics. To obtain richer features, we can use ML to learn a representation from the audio. These learned features can often improve performance for a given MIR task.

In order to learn interesting musical representations, it is important to consider the particular aspects of music audio when building learning models. Given the temporal and spectral structure of music audio, deep and multi-scale representations are particularly well suited to represent music. This thesis focuses on learning representations from music audio. Deep and multi-scale models that improve the state-of-the-art for tasks such as instrument recognition, genre recognition and automatic annotation are presented.

Keywords: machine learning, music information retrieval, music audio analysis, automatic annotation, deep learning, multiscale learning.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

Par souci de clarté et de cohérence vis-à-vis la littérature scientifique, l'acronyme anglais est conservé dans la majorité des cas. Pour les nombreuses traductions de termes spécifiques au domaines présentes dans le texte, le terme original en anglais est spécifié en *italique*.

AM	Apprentissage machine	*Machine Learning*
AUC	Aire sous la courbe ROC	*Area under the ROC curve*
CD	Divergence Contrastive	*Contrastive Divergence*
CDBN	DBN à Convolution	*Convolutional Deep Belief Network*
DBN		*Deep Belief Network*
DCT	Transformée cosinusoïdale discrète	*Discrete Cosine Transform*
DFT	Transformation de Fourier Discrète	*Discrete Fourier Transform*
EM	Espérance-Maximisation	*Expectation-Maximization*
FFT	Transformation de Fourier Rapide	*Fast Fourier Transform*
GMM	Modèle de mixture de Gaussiennes	*Gaussian mixture model*
HMM	Modèle de Markov Caché	*Hidden Markov Model*
KNN	K plus proches voisins	*K-nearest neighbor*
LDA	Analyse discriminante linéaire	*Linear Discriminant Analysis*
MDS	Échelonnage Multi-dimensionnel	*Multidimensional Scaling*
MFCC	Coefficients Cepstraux de Fréquence Mel	*Mel-Frequency Cepstral Coefficients*

MIDI		*Musical Instrument Digital Interface*
MIR	Recherche d'information musicale	*Music Information Retrieval*
MLP	Perceptron Multi-couches	*Multilayer perceptron*
MTSL	Apprentissage à multiples échelles temporelles	*Multi-Time-Scale Learning*
PCA	Analyse en Composantes Principales	*Principal Component Analysis*
PCM	Modulation d'Impulsion Codée	*Pulse Code Modulation*
PFC	Classifieur de caractéristiques sous-échantillonées	*Pooled Features Calssifier*
PMSC	Composantes Principales du Spectre Mel	*Principal Mel-spectrum components*
RBM	Machine de Boltzmann restreinte	*Restricted Boltzmann Machine*
ROC	Caractéristique de fonctionnement du récepteur	*Receiver operating characteristic*
SVM	Machine à vecteurs de support	*Support Vector Machine*
VQ	Quantification vectorielle	*Vector quantization*

CHAPITRE 1

INTRODUCTION

L'avènement du mp3 et de la numérisation de la musique a apporté de grands changements au domaine musical. Les moyens de production, de distribution et de consommation de la musique se sont vus révolus au cours des dernières décennies. Les coûts de production ont drastiquement diminué, permettant à des particuliers d'avoir accès à des moyens d'enregistrement professionnel abordables. Cela a mené à une démocratisation de la musique, multipliant le nombre de nouveaux artistes indépendants sur le marché. L'internet permet maintenant de distribuer la musique sans médium physique, permettant ainsi une mondialisation du marché de la musique. Finalement on voit une tendance vers l'écoute de la musique en ligne soit par des radios internet, ou des système de stockage « dans le nuage » (*in the cloud*). Tout cela a pour effet de drastiquement augmenter la quantité de musique disponible pour un auditeur. Il n'est plus rare de voir des collections de musique personnelles équivalentes à plusieurs semaines d'écoute continues. Dans le cas de radios internet, la bibliothèque disponible à l'utilisateur devient virtuellement infinie. Le problème devient alors de savoir quoi écouter, parmi l'immensité de choix disponible. Devant l'immensité de toute la musique disponible, les systèmes se fiant sur des auditeurs experts, limités en temps et en ressources, ne peuvent qu'analyser une faible partie de cet océan d'oeuvres musicales, en se concentrant principalement sur les artistes les plus populaires.

Heureusement, la technologie moderne nous permet de construire des systèmes capables d'analyser automatiquement certains aspects de la musique. Le domaine de la recherche d'information musicale (*Music Information Retrieval* ou *MIR*) se spécialise dans la conception de tels systèmes. Ces systèmes permettent, entre autres, de faciliter la recherche et la classification de documents musicaux, d'obtenir des mesures de similarités et de concevoir de nouveaux outils pour les musiciens. Le MIR utilise l'apprentissage machine (AM) comme outil afin de résoudre un grand éventail de tâches.

Les données musicales présentent plusieurs défis pour l'AM. Premièrement, la mu-

sique présente des dépendances temporelles à long terme. Par exemple, dans une pièce, un même thème peut être répété à plusieurs minutes d'intervalle. Plusieurs modèles d'AM échouent à apprendre des dépendances à long terme à partir de données séquentielles. Ensuite, les données musicales sont souvent à très haute dimension. On doit donc utiliser des méthodes d'extraction de caractéristiques ou de réduction de dimensionalité, selon la tâche à résoudre. Également, étant donné la subjectivité de l'analyse musicale on se voit souvent confronté à l'absence de vérité absolue (*ground thruth*) dans plusieurs tâches. Par exemple, dans le contexte de reconnaissance de genre, une même pièce pourrait être étiquetée comme du jazz ou du blues selon différents experts. Enfin, la musique véhicule plusieurs concepts de haut niveau expressifs et sémantiques. Ce sont ces concepts abstraits qui amènent une réaction affective de l'auditeur envers la musique. Il serait donc utile de pouvoir mettre en évidence ces aspects de la musique.

Au coeur des défis énumérés ci-haut se pose le problème de la représentation de la musique. C'est-à-dire, comment peut-on mettre en évidence les caractéristiques importantes de la musique dans un format adéquat pour effectuer une tâche donnée. Bien sûr, les caractéristiques pertinentes varient selon la tâche à effectuer. L'approche traditionnelle de MIR consiste à extraire un ensemble de caractéristiques obtenues à l'aide d'heuristiques de traitement de signal basées sur la connaissance du domaine. Malheureusement, souvent, ce genre de méthode d'extraction de caractéristiques laisse tomber des informations importantes contenues dans la musique, soit parce qu'il n'existe pas de technique connue pour extraire un concept donné, ou bien parce que le concept est tout simplement trop abstrait pour être bien cerné. Une approche pour obtenir des caractéristiques plus riches repose sur l'idée d'apprendre une représentation de la musique à partir de l'audio musical. Pour ce faire, l'utilisation de méthodes supervisées et non supervisées ont déjà montré un certain potentiel. L'apprentissage de telles représentations peut s'appliquer dans un grand nombre de tâches de MIR. De plus, des avancées sur les représentations musicales peuvent également se transférer dans d'autres domaines où l'on rencontre des données séquentielles, à haute dimension, présentant de hauts niveaux d'abstraction, tels que la vidéo, la bourse, ou les jeux vidéos.

Cette thèse porte sur l'amélioration de représentations de l'audio musical grâce à

l'AM dans le cadre du MIR. Nous verrons que l'apprentissage de représentations profondes et multi-échelles permet d'obtenir de meilleures performances pour plusieurs tâches de MIR. La thèse est divisée comme suit. Premièrement, nous établirons le contexte théorique de l'AM au chapitre 2 et du MIR au chapitre 3. Ensuite, nous présenterons les articles inclus dans cette thèse au chapitre 4. Les chapitres 5 à 8 sont constitués de travaux publiés portant sur les représentations profondes et multi-échelles de l'audio musical. Finalement, au chapitre 9, nous conclurons en résumant les résultats importants.

CHAPITRE 2

APPRENTISSAGE MACHINE

L'apprentissage machine (AM) est un sous-domaine de l'intelligence artificielle qui consiste à élaborer des algorithmes capables d'apprendre, à partir d'exemples. Il ne s'agit pas ici d'apprendre les exemples par coeur, mais plutôt de bien généraliser à partir de ces exemples, afin d'effectuer une certaine tâche. Pour des tâches complexes telles que la reconnaissance de chiffres et de caractères ainsi que la reconnaissance de la voix, par exemple, l'AM peut arriver à des taux d'erreurs beaucoup plus bas que les algorithmes classiques.

2.1 Cadres d'apprentissage

L'AM peut être appliqué à une grande variété de tâches et de données. Selon le cas, différents cadres d'apprentissage peuvent s'appliquer.

2.1.1 Apprentissage supervisé

Dans le cadre supervisé, les données $z_i = (x_i, y_i) \in \mathbb{R}^m \times \mathbb{R}^n$ sont formées d'une entrée $x_i \in \mathbb{R}^m$ et d'une cible, ou étiquette, $y_i \in \mathbb{R}^n$. Le but est alors d'apprendre une fonction $f(x_i) \in \mathbb{R}^n$ capable de prédire la cible. La nature de la cible définit le genre de problème à résoudre.

Dans le cas où la cible est discrète, on parle alors d'un problème de **classification**. Le cas le plus simple est la classification binaire, où il n'y a que deux classes à séparer. Dans ce cas, on peut représenter les deux classes par 0 et 1. S'il y a plus de deux classes, une bonne façon de représenter la cible est par un vecteur *one-hot* (i.e. un vecteur où tous les éléments sont nuls sauf un qui est égal à 1). Il se peut aussi qu'un exemple d'entrée puisse appartenir à plusieurs classes en même temps, dans ce cas on parlera d'**étiquetage** plutôt que de classification.

Si la cible est continue, on parle alors d'un problème de **régression**. Dans ce cas, le

but de l'apprentissage est de trouver un fonction continue qui prédit y étant donné les entrées x.

2.1.2 Apprentissage non-supervisé

Dans le cas non-supervisé, les données $z_i = x_i \in \mathbb{R}^m$ n'ont pas de cible. L'apprentissage non-supervisé peut nous permettre d'estimer la densité de probabilité des entrées, d'effectuer une réduction de dimensionalité, de trouver un partitionnement des données ou de trouver une nouvelle représentation des données.

L**'estimation de densité de probabilité** cherche à trouver la distribution de probabilité sous-jacente aux données. De modéliser la distribution des entrées peut être utile pour évaluer si un nouveau point de donnée vient de la même distribution que l'ensemble d'entraînement. Certains modèles dits génératifs sont capables de générer de nouveaux points de données à partir la densité de probabilité. On appelle ce processus échantillonnage.

La **réduction de dimensionalité**, explorée plus en détails à la section 2.4, est utile dans de nombreux cas. Entre autres, elle peut permettre de visualiser des données à haute dimension, d'extraire des caractéristiques, d'accélérer l'apprentissage ou de réduire la quantité de mémoire requise.

Le **partitionnement non-supervisé** nous permet de déterminer s'il y a des groupement de points dans l'espace d'entrée, où ils sont, et combien il y en a. Cela peut être très utile pour avoir une idée de la structure des entrées. De plus, en analysant la distribution des entrées pour chaque partition, on peut déterminer qu'est-ce qui distingue les différents groupes, ce qui nous donne encore plus d'information sur la structure des données.

L'**apprentissage de représentation** est le cas qui nous intéresse le plus dans ce document. La représentation apprise peut être de dimension moindre, plus grande ou égale à la dimension d'entrée. La réduction de dimensionalité est donc un cas particulier de l'apprentissage de représentation. Un but de l'apprentissage de représentation peut être de trouver un espace à partir duquel il sera plus facile de résoudre une tâche donnée. On peut aussi voir l'apprentissage de représentation comme une extraction de caracté-

ristiques.

2.1.3 Apprentissage semi-supervisé

Il arrive parfois qu'on puisse avoir beaucoup de données d'entrée, mais qu'il soit coûteux d'obtenir des cibles pour ces données, ou bien qu'il soit possible d'avoir des cibles pour seulement un sous-échantillon de l'ensemble. Cela arrive souvent dans le cas de l'audio musical où il est facile de trouver des bases de données immenses d'exemples non-étiquetés en grande quantité. Par contre, selon la tâche à résoudre, il se peut que l'étiquetage nécessite le travail d'un expert qui devra annoter à la main chaque exemple.

Il est possible de tirer parti de cette situation grâce au cadre semi-supervisé. On suppose dans ce cas qu'on peut tirer de l'information supplémentaire à partir des données non-étiquetées, ce qui améliorera la performance de la tâche supervisée.

2.2 La généralisation et la minimisation de l'erreur empirique

La plupart des algorithmes d'apprentissages peuvent être formulés comme un problème de minimisation de coût. Soit θ les paramètres d'un algorithme d'apprentissage et $D = (z_1, z_2, ..., z_N)$ un ensemble de N données. On choisit une fonction de coût $L(z_i, \theta)$ selon la tâche à accomplir. L'erreur empirique sur D est alors définie par :

$$\mathscr{L} = \frac{1}{N} \sum_{i=1}^{N} L(z_i, \theta). \tag{2.1}$$

L'algorithme cherche à modifier ses paramètres θ afin de minimiser $\mathscr{L}$ sur l'ensemble d'entraînement. Une façon de faire cela est de mettre à jour les paramètres en suivant le gradient $-\frac{\delta \mathscr{L}}{\delta \theta}$. Il y a toutefois un problème avec la minimisation de l'erreur empirique. Une faible erreur empirique sur l'**ensemble d'entraînement** ne signifie pas nécessairement une faible erreur de généralisation sur la vraie distribution des entrées. Par conséquent, afin d'obtenir une estimation non-biaisée de l'erreur de généralisation, on calcule l'erreur empirique sur un **ensemble de test** qui n'a pas été utilisé durant l'entraînement.

En minimisant l'erreur empirique sur l'ensemble d'entraînement, un modèle ayant assez de degrés de liberté (ou de capacité) aura tendance à apprendre par coeur l'ensemble de données. C'est-à-dire que l'algorithme apprendra des particularité de l'ensemble d'entraînement qui ne sont pas des caractéristiques de sa distribution sous-jacente. C'est ce qu'on appelle le **sur-apprentissage**. Le but d'un algorithme d'apprentissage étant de **généraliser** sur de nouvelles données tirées d'une même distribution, on cherche à éviter ce problème.

Une solution au problème de sur-apprentissage est de limiter la capacité du modèle. Limiter la capacité d'un algorithme d'apprentissage oblige le modèle à apprendre seulement les caractéristiques les plus représentatives de l'ensemble de données. C'est ce qu'on appelle la **régularisation.** Selon le modèle, la régularisation peut prendre plusieurs formes (e.g. limiter le nombre ou la norme des paramètres, arrêt précoce de l'entraînement (*early-stopping*), etc.). La plupart du temps, un ou plusieurs paramètres λ sont nécessaires pour définir le niveau de régularisation. On appelle ces paramètres les hyper-paramètres du modèle. Souvent, ces hyper-paramètres doivent être optimisés par essais et erreurs. Pour l'optimisation des hyper-paramètres, on utilise un autre ensemble de données indépendant de l'ensemble d'entraînement appelé l**'ensemble de validation**. On entraîne alors plusieurs modèles ayant des hyper-paramètres différents sur l'ensemble d'entraînement, et on sélectionne celui qui obtient l'erreur empirique la plus basse sur l'ensemble de validation.

2.3 Quelques modèles d'apprentissage machine

Dans cette section, je décrirai brièvement quelques algorithmes d'apprentissage. Pour une liste plus complète et plus de détails, consultez [14].

2.3.1 Classifieurs

La classification, telle que décrite à la section 2.1.1, nous permet de discriminer entre deux ou plusieurs classes.

Lorsque les données sont linéairement séparables (i.e. : peuvent être séparées par des

surfaces de décision linéaires), un simple classifieur linéaire peut aisément effectuer le travail de classification. Le perceptron et la régression logistique sont deux exemples de classifieurs linéaires.

Toutefois, dans le cas où les classes ne sont pas linéairement séparables dans l'espace des entrées, il peut être préférable d'utiliser un classifieur non-linéaire. Les machines à vecteurs de support (*Support vector machines* ou SVMs) à noyau [15, 23] sont un exemple de classifieur non-linéaire très largement utilisé. Le SVM utilise le truc du noyau pour projeter les données dans un espace à haute dimension avant d'effectuer une classification linéaire (figure 2.1). Cela résulte en une classification non-linéaire dans l'espace des entrées. Le SVM présente un problème d'optimisation convexe quadratique, ce qui garantit de trouver le minimum global, mais au coût d'un temps de calcul de l'ordre de N^2, ce qui devient prohibitif pour de grands ensembles de données.

On peut aussi obtenir un classifieur non-linéaire à l'aide du perceptron multi-couche (*Multilayer Perceptron*, ou MLP). Le MLP est un approximateur universel. C'est-à-dire qu'avec assez de capacité, le MLP peut, en théorie, approximer n'importe quelle fonction non-linéaire. L'optimisation du MLP se fait par rétropropagation du gradient. Cette optimisation n'est pas convexe, ce qui amène le risque de tomber dans un minimum local. L'optimisation d'un MLP est toutefois plus rapide que celle d'un SVM pour de larges ensembles de données.

La technique des K plus proches voisins (*K nearest neighbors* ou KNN) utilise l'information locale autour d'un point afin de lui assigner une classe. Il suffit de trouver les K points les plus proches dans l'espace d'entrée, et de choisir la classe qui revient le plus fréquemment dans ce sous-ensemble de points. Cette méthode présente l'avantage d'avoir un seul paramètre à optimiser (K), et de ne nécessiter aucune optimisation préalable. Pour classifier un nouveau point, il faut toutefois calculer la distance entre ce point et tous les autres points de l'ensemble de données. Cela peut devenir computationnellement coûteux lorsque l'ensemble de données est grand.

Le *boosting* est une autre technique de classification non-linéaire. Cette technique utilise une multitude de classifieurs faibles afin de construire un classifieur final. Un classifieur faible est tout simplement un classifieur qui obtient une meilleure classifica-

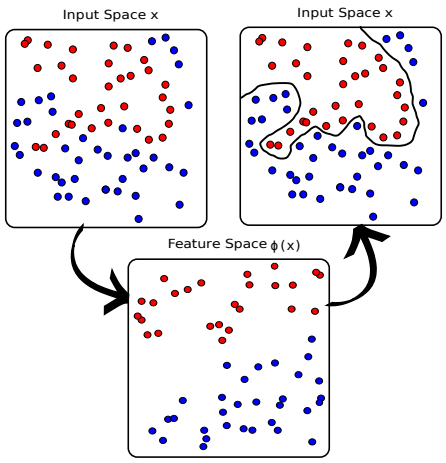


Figure 2.1 – Schéma du fonctionnement d'un SVM

tion que la chance pour discriminer deux classes. À chaque itération, les exemples mal classifiés par le classifieur final se voient attribuer une plus grande importance pour le prochain classifieur faible.

2.3.2 Partitionnement non-supervisé

Avec un ensemble de données non-étiqueté, on peut utiliser la distribution de densité empirique des données pour en déduire un partitionnement. La méthode des K-moyennes est probablement la plus simple des techniques de partitionnement non-supervisé. On commence par initialiser K points représentant les K moyennes au hasard dans l'espace d'entrée. Ensuite, dans une première phase, on assigne à chaque donnée de l'ensemble la moyenne la plus près. Dans une deuxième phase, on recalcule les K moyennes en fonction des points de donnée qui lui sont attribués. On alterne ainsi les deux phases jusqu'à la convergence. Cette méthode de convergence s'appelle espérance-maximisation (*expectation-maximisation* ou EM).

Le modèle de mixture de gaussiennes (*Gaussian Mixture Model* ou GMM) est une version plus évoluée de K-moyennes. Ici, au lieu d'utiliser seulement des moyennes, on utilise des gaussiennes pour décrire la distribution des données. Le GMM possède donc plus de paramètres à optimiser que les K-moyennes. Il permet cependant d'obtenir une modélisation plus raffinée de la distribution de densité des données. On peut également

utiliser l'EM pour entraîner un GMM.

2.4 Données à haute dimension et réduction de dimensionalité

Lorsqu'on travaille avec de l'audio musical, il est important de considérer le fait que l'audio est une représentation à très haute dimension. Par exemple, si on considère 5 secondes de musique enregistrée à une fréquence d'échantillonnage de 44100 Hz, on a besoin de $44100 * 5 = 220500$ échantillons pour représenter cet extrait. Il est problématique de travailler avec des extraits audios le moindrement long si on n'essaie pas de réduire la dimensionalité de la représentation. Avec de telles données à haute dimensionalité, on se frappe bien sûr à des problèmes de ressources computationnelles limitées, mais aussi au phénomène du fléau de dimensionalité.

Le fléau de dimensionalité est dû au fait que lorsqu'on ajoute des dimensions à un espace, le volume augmente exponentiellement. Cela a pour conséquence que le nombre de points nécessaire pour échantillonner l'espace augmente aussi exponentiellement. Par exemple, pour échantillonner un cube unitaire à 3 dimensions avec une distance de 0.1 entre les points, on a besoin de 10^3 points. Pour échantillonner un hyper-cube unitaire à 10 dimensions, on aura alors besoin de 10^{10} points, soit 10 millions fois plus de points. Cela est un problème pour l'apprentissage machine étant donné que le nombre d'échantillons et le temps de calcul sont limités.

Un autre aspect du fléau de dimensionalité est que, dans un espace à haute dimension, les points ont tendance à être loin du centre, car une grande partie du volume de l'espace se situe "dans les coins". Pour illustrer ce fait, considérons une hyper-sphère de rayon r et un hyper-cube de côté $2r$ dans un espace à d dimensions. L'hyper-sphère a un volume de :

$$\frac{2\pi^{d/2}}{d\Gamma(d/2)} r^d \tag{2.2}$$

où Γ est la fonction Gamma, l'extension de la factorielle au domaine des complexes (et donc des réels). L'hyper-cube a quant à lui un volume de :

$$(2r)^d \tag{2.3}$$

Le rapport entre le volume de l'hyper-sphère et l'hyper-cube est donc :

$$\frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \tag{2.4}$$

Ce rapport tend vers 0 lorsque $d \to \infty$. Cela signifie que la portion d'espace occupée par l'hyper-sphère devient négligeable par rapport au volume de l'hyper-cube. Donc, la grande partie de l'espace se retrouve "dans les coins" (un hyper-cube à d-dimensions possède 2^d coins). Cela a pour conséquence que, dans un espace à haute dimension, les points ont tous tendance à être autant éloignés les uns des autres, et la mesure de distance euclidienne devient moins informative.

Pour contourner le problème du fléau de dimensionalité, il existe plusieurs techniques de réduction de dimensionalité qui projettent les données dans un sous-espace de l'espace d'entrée dans lequel les données sont mieux distribuées. Si on projette les données en 2 ou 3 dimensions, on peut alors parler de visualisation. Voici quelques-unes des techniques de réduction de dimensionalité.

2.4.1 Analyse en composantes principales

L'analyse en composantes principales (*Principal Component Analysis* ou PCA) est une méthode de réduction de dimensionalité utilisée dans une multitude de domaines. L'algorithme de PCA représente un ensemble de données dans un nouveau système d'axes dans lequel chaque dimension est décorrélée. Cette décorrélation des variables nous permet de réduire la représentation à la dimension voulue en choisissant les axes pour lesquels la variance des données est la plus grande. Par conséquent, la PCA nous donne une compression linéaire qui explique le mieux la variance dans les données.

2.4.2 Échelonnage Multi-Dimensionnel

L'échelonnage multi-dimensionnel (*Multi-Dimensional Scaling* ou MDS) nous permet de construire un espace à partir d'une matrice de distances. Cela peut être pratique dans le cas où on a une distance entre différents concepts, mais qu'on ne connaît pas le nombre de dimensions, et comment définir ces dimensions. Un exemple d'application de

la MDS est la construction d'un espace de timbre musical. Le timbre musical peut être défini comme tout ce qui n'est pas hauteur et amplitude d'une note. À l'oreille, on peut juger si deux timbres sont similaires ou différents, mais il est toutefois difficile de définir explicitement quelles caractéristiques utiliser pour définir cette distance. Avec la MDS, on a pu construire un espace de timbres, à partir de la distance psycho-acoustique perçue par les participants à l'expérience [80, 102]. En analysant l'espace à 3 dimensions ainsi obtenu, on a pu nommer les trois critères les plus important dans la caractérisation du timbre musical.

2.4.3 t-SNE

Un autre aspect qui peut être intéressant à visualiser est comment des données en haute dimension se regroupent ensemble. Toutefois, les réductions de dimensionalité linéraires telles que PCA, ne préservent pas bien ces structure locales. L'algorithme t-SNE [97] est spécialement conçu dans le but de préserver la structure locale de l'espace à haute dimension. Cela permet de visualiser si un ensemble de données à haute dimension présente des groupements, et quels sont-ils.

2.5 Apprentissage profond

Jusqu'à tout récemment, la grande majorité de la recherche en AM se concentrait sur des architectures peu profondes. C'est-à-dire, des architectures comportant une composition d'au plus trois transformations non-linéaires dans la fonction apprise [4]. Celles-ci ont parfois l'avantage de présenter une optimisation convexe. De plus, en théorie, avec assez de données, de capacité et de temps de calcul, certains modèles peu profonds peuvent représenter n'importe quelle fonction complexe. Toutefois, ces architectures présentent des lacunes au niveau du pouvoir de représentation et de la généralisation non-locale. Le pouvoir de représentation restreint des architectures peu profondes a comme conséquence d'utiliser plus de ressources que nécessaire pour représenter un fonction donnée. Bengio & LeCun (2007) [5] présentent des fonctions pour lesquelles la représentation par un réseau de neurones à une couche cachée nécessite exponentielle-

ment plus de neurones que par un réseau profond.

En général, une architecture profonde est composée de plusieurs couches de modules non-linéaires paramétrisés. Ce sont ces paramètres qui sont soumis à l'apprentissage. Les architectures profondes possèdent un meilleur pouvoir de représentation que les architectures peu profondes. Chaque couche permet une représentation de plus haut niveau que la précédente. On espère donc apprendre une représentation abstraite des données, dans laquelle la tâche à résoudre sera plus facile. Une généralisation locale à partir de cette représentation permet une généralisation non-locale dans l'espace des entrées.

À l'exception des réseaux de neurones convolutionnels (section 2.5.1), les architectures profondes ont longtemps été impopulaires à cause du manque de technique adéquate d'optimisation. La fonction de coût de tels modèles étant hautement non-convexe, les techniques de descente de gradient ont donc tendance à tomber dans des minima locaux, souvent loins de la solution optimale. Ce problème est peut-être dû en partie à la saturation causée par les fonctions d'activations sigmoïde et tangente hyperbolique [41].

Hinton et al. (2006) [52] ont proposé une technique de pré-entraînement non-supervisé basée sur des Machines de Boltzmann Restreintes (*Restricted Boltzmann Machines* ou RBMs) qui permet de préparer le réseau avant d'utiliser la descente de gradient. Depuis ce temps, d'autres techniques de pré-entraînement ont été développées et appliquées à différentes architectures profondes [6, 52, 100]. Celles-ci reposent sur le même principe d'un pré-entraînement vorace couche par couche.

Bien qu'on ait démontré que le pré-entraînement d'un réseau profond fonctionne bien en pratique, il y a encore beaucoup de mystère quant à pourquoi cela fonctionne bien. Ehran et al. (2010) [31] ont effectué plusieurs expériences afin de mettre un peu de lumière sur cette question.

2.5.1 Réseaux de neurones convolutionnels

Les premières architectures profondes fonctionnelles à faire leur apparition ont été les Réseaux de Neurones Convolutionnels (*Convolutional Neural Networks* ou CNNs) [63, 64, 101]. Ces modèles ont démontré leur valeur dans le domaine de la vision et de la reconnaissance de parole. Nous décrirons ici le CNN appliqué sur des données d'entrées

matricielles $N \times N$ pouvant représenter, par exemple, une image. Il est toutefois possible d'adapter ce modèle à des entrées vectorielles ou tensorielles.

Dans un CNN, on retrouve deux types de couches en alternance : une couche de convolution et une couche de sous-échantillonage (*pooling*). La couche de convolution est formée d'un certain nombre de filtres ayant un champ de perception limité (e.g. des petits carrés $n \times n$ tel que $n < N$). Ces filtres agissent en tant que détecteurs de caractéristiques. Ils sont définis par des poids qui sont ajustables pendant l'apprentissage. On effectue une convolution de ces filtres sur l'entrée pour obtenir les activations de la couche. On peut voir la convolution des filtres comme un partage de poids entre les unités d'un même filtre appliqué à différents endroits. Les activations sont ensuite envoyées à la couche de sous-échantillonage. Celle-ci, également formée d'unités ayant un champ de perception limité, réduit la dimension de la couche précédente grâce à une fonction de sous-échantillonnage (typiquement la moyenne ou le maximum des activations dans le champ perceptif). La prochaine couche effectue ensuite une convolution sur les sorties de la couche de sous-échantillonnage, et ainsi de suite. Après la dernière couche de sous-échantillonnage, on ajoute une couche de neurones de sortie, nous permettant ainsi d'effectuer l'entraînement supervisé en rétropropageant le gradient.

L'alternance entre convolution et sous-échantillonnage donne une structure pyramidale au CNN. Ainsi, les couches supérieures représentent des caractéristiques de plus en plus globales de l'entrée car leur champ perceptif, bien que de faible dimensionalité, correspond à une plus grande partie de l'entrée. De plus, le partage de poids permet de limiter le nombre de paramètres du modèle, ce qui facilite l'apprentissage.

2.5.2 DBN

Le modèle de *Deep Belief Network* (DBN) [52] est un réseau de neurones profond, probabiliste et génératif. Un DBN est constitué de plusieurs Machines de Boltzmann Restreintes (*Restricted Boltzmann Machines* ou RBMs) empilées les unes sur les autres.

Les machines de Boltzmann [54] sont des modèles probabilistes génératifs qui tentent de modéliser la distribution de probabilité des entrées. Elles sont formées d'un réseau d'unités (neurones) visibles et cachées reliées par des connections non-dirigées avec

poids. Les Machines de Boltzmann Restreintes sont des machines de Boltzmann avec des restrictions sur les connections. Elles possèdent deux couches : une couche visible et une couche cachée. Chaque unité de chaque couche est reliée à toutes les unités de l'autre couche, mais il n'y a aucune connection entre les unités d'une même couche. Le fait d'interdire les connections sur une même couche a pour conséquence que les unités d'une même couche deviennent conditionnellement indépendantes si l'autre couche est connue. Cela permet un échantillonnage de Gibbs très efficace. L'algorithme d'apprentissage non-supervisé de Divergence Contrastive *(Contrastive Divergence* ou CD) décrit par Hinton et al. (2006) utilise ce fait pour obtenir une méthode très rapide pour entraîner une RBM.

Un DBN est construit par une superposition de plusieurs RBMs. La couche d'entrée du DBN est la couche visible de la première RBM. La couche cachée de la première RBM devient la couche visible de la deuxième RBM, et ainsi de suite.

Lors du pré-entraînement, on entraîne les RBMs une par une de manière vorace. C'est-à-dire qu'on entraîne la première RBM avec les entrées de l'ensemble de données, puis on entraîne le second RBM avec la couche cachée du premier RBM, etc.

Pour effectuer de l'apprentissage supervisé, aussi appelé ajustement fin (*fine-tuning*), on ajoute après le dernier RBM une couche de sortie qui tentera de prédire les cibles de l'ensemble d'entraînement. L'ajustement fin consistera à rétropropager l'erreur jusqu'à la première couche du DBN, comme dans un réseau de neurones traditionnel.

2.5.3 DBN convolutionnels

En combinant les concepts de CNN, RBM et de DBN, on peu obtenir un réseau profond convolutionnel.

Desjardins et al. [25] présentent une adaptation du RBM permettant d'opérer de manière convolutionnelle. Pour une tâche de vision simple, le CRBM tend à converger plus rapidement et obtient une meilleure mesure de vraisemblance que des RBMs conventionnels.

Le DBN convolutionnel (*Convolutional Deep Belief Network* ou CDBN) a été introduit par Lee et al. (2009) [66]. Cet article, démontre le pouvoir de représentation

de ce modèle dans le domaine de la vision. Les CDBNs, grâce à leur structure, apprennent naturellement une représentation hiérarchique des données, et ce, de manière non-supervisée. De plus, les CDBNs sont génératifs. Cela signifie qu'on peut échantillonner à partir du modèle pour obtenir un nouvel exemple d'entrée ou pour compléter une entrée dont certaines données seraient manquantes. Le CDBN a été appliqué à l'audio dans [65]. Toutefois, celui-ci a été appliqué à seulement des fenêtres de temps relativement courtes (180 ms). Il serait intéressant de voir quelle représentation serait apprise par le CDBN sur des périodes de temps plus longues.

CHAPITRE 3

RECHERCHE D'INFORMATION MUSICALE

Le domaine de la recherche d'information musicale est un point de rencontre pour plusieurs domaines tels que l'apprentissage machine, le génie électrique, la musicologie et la bibliothéconomie. Dans ce document, on se consacrera d'avantage à l'aspect apprentissage machine du MIR, mais il est tout de même indispensable de faire un survol des techniques d'analyse de signal nécessaires au pré-traitement des données afin de bien comprendre les enjeux de la recherche en MIR.

Puisque cette thèse traite de l'apprentissage de représentations musicales, une description des représentations et des caractéristiques les plus couramment utilisées dans le domaine du MIR débutera ce chapitre. Ensuite, différentes tâches de MIR seront définies. Finalement, une discussion portant sur les défis rencontrés dans l'analyse d'audio musical concluera le chapitre.

3.1 Représentations symboliques

Il existe une multitude de représentations symboliques de la musique (partition, MIDI, abc, musicXML, tablatures, etc.). La notation symbolique la plus universelle, dans le monde occidental, est évidemment la portée musicale, ou partition. C'est avec cette représentation que s'est transmise la musique occidentale au cours des derniers siècles. La partition peut être plus ou moins précise, selon le type de musique. Toutefois, peu importe la quantité de détails présents dans une partition, il y a toujours de l'information implicite qui nécessite une connaissance musicale pour jouer cette pièce. La partition permet également un certain degré de liberté face à comment interpréter la partition. Par exemple, le terme *forte* signifiant au musicien de jouer fort ne donne aucune précision sur le niveau exact de décibels à atteindre. Le musicien doit interpréter la partition et décider de la force de la note selon plusieurs critères musicaux (force des notes précédentes, position de la note dans la phrase, style musical, expressivité, intention du

compositeur, etc.). Chaque musicien va interpréter à sa manière les symboles, et trouver la manière qui lui convient de jouer le morceau.

Une autre notation symbolique plus récente est le format MIDI. Le MIDI est utilisé par les musiciens modernes en tant que contrôleur pour produire de la musique avec des instruments virtuels. Un fichier MIDI contient des informations sur une liste d'événements (*noteOn, noteOffs, pitch bend, pedal*, etc.) dans le temps. Une note MIDI peut être décrite par sa position dans le temps, sa hauteur, sa vélocité et sa durée. De cette manière, on peut visualiser les notes d'un fichier midi à l'aide d'un *pianoroll* (figure 3.1). Le format MIDI permet donc de noter très précisément le moment et la force à laquelle la note doit être jouée. Par contre, il y a peu d'information sur la qualité du timbre d'une note. Par exemple, dans un solo de violon, le timbre produit par un musicien variera de façon continue afin de traduire l'expressivité du morceau. Il est théoriquement possible de contrôler certains paramètres du timbre à partir du MIDI, selon l'instrument virtuel utilisé, mais ces contrôles sont trop souvent peu intuitifs.

3.2 Représentations de l'audio

Physiquement, le son est une variation de la pression d'air. Il peut-être capté à l'aide d'une membrane comme le tympan ou un microphone. Un son enregistré à l'aide de microphones est traduit en signal électrique. Ce signal est ensuite échantillonné afin d'obtenir une représentation digitale.

Pour obtenir des caractéristiques du son, on utilisera des techniques de traitement de signal. Le traitement de signal est un domaine du génie électrique qui se spécialise dans l'analyse de données séquentielles. Ces données peuvent être séquentielles dans le temps (e.g. audio, ondes radio, électro-cardiogramme, etc.) ou dans l'espace (e.g. images) ou même les deux à la fois (e.g. vidéos). L'analyse de signal nous permet de manipuler le son afin d'obtenir différentes représentations. Chacune de ces représentations présentent ses avantages et ses inconvénients.

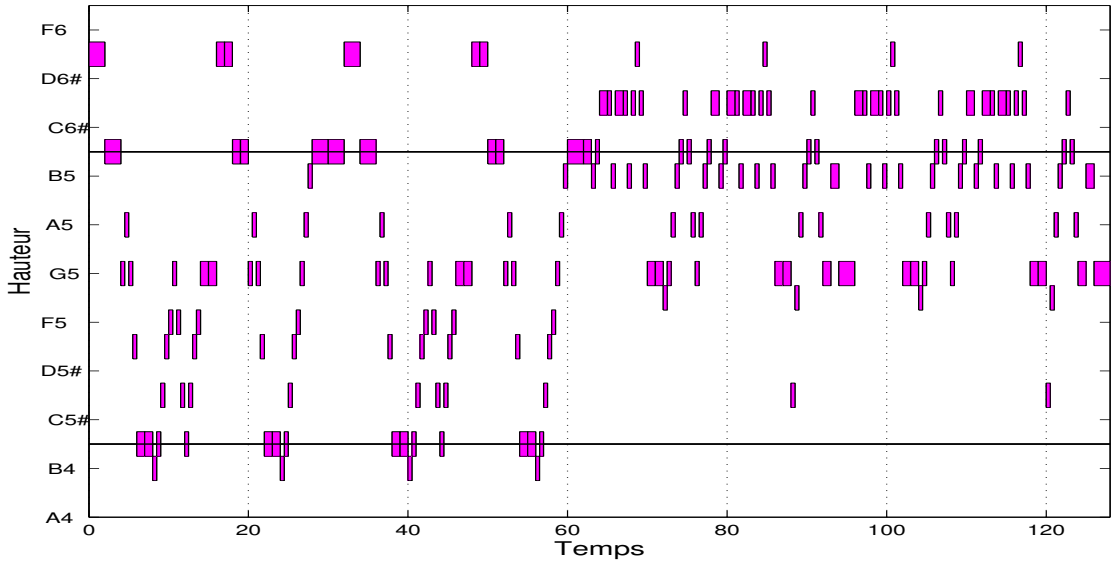


Figure 3.1 – Exemple de *pianoroll*

3.2.1 Domaine temporel

La modulation d'impulsion codée (*Pulse Code Modulation* ou PCM), aussi connu connu comme le format WAV (*waveform*) , est l'encodage numérique de base pour l'audio. Dans ce format, le signal audio est décrit par une séquence de valeurs représentant l'amplitude du signal dans le temps. La qualité d'un encodage PCM dépend de la fréquence d'échantillonnage f_e et du nombre de bits de précision b. La fréquence d'échantillonnage définit l'intervalle de temps entre deux mesures d'amplitude. Le nombre de bits correspond à la précision d'une mesure d'amplitude.

Le théorème d'échantillonnage de Nyquist-Shannon stipule que, dans un échantillonnage discret d'un signal continu, les fréquences f et $f_e - f$ sont indistingables, c'est ce qu'on appelle le problème de repliement (*aliasing*). Pour éviter ce problème, il est important de filtrer les fréquences plus hautes que la fréquence de Nyquist $f_N = \frac{f_e}{2}$. La bande de fréquences audible par l'oreille humaine se situe entre 20 Hz et 20 kHz, et varie en fonction de l'âge de l'individu. Un bon encodage audio nécessite donc une fréquence d'échantillonnage supérieure à 40 kHz. Par exemple, les disques compacts ont une fréquence d'échantillonnage de 44,1 kHz et une précision de 16 bits. On trouve maintenant de plus en plus d'enregistrements numériques à 48 kHz et 32 bits.

Il est possible d'extraire plusieurs caractéristiques temporelles du signal telles que l'enveloppe d'amplitude, le taux de passage à zéro (*zero-crossing rate*) et l'auto-corrélation à partir de la représentation PCM.

3.2.2 Domaine fréquentiel

La transformée de Fourier est certainement une des techniques d'analyse de fonction les plus anciennes, et aussi une des plus utiles. Elle permet de représenter une fonction dépendante du temps dans l'espace des fréquences.

La transformée de Fourier F d'une fonction f est définie par la formule suivante :

$$F(\nu) = \int_{-\infty}^{\infty} f(t)\, e^{-i2\pi t\nu}\, dt \tag{3.1}$$

où t représente le temps et ν la fréquence. $F(\nu)$ est aussi appelé le spectre du signal. Cette transformation est inversible, et son inverse est :

$$f(t) = \int_{-\infty}^{\infty} F(\nu)\, e^{i2\pi t\nu}\, d\nu. \tag{3.2}$$

Cette définition est valable lorsque $f(t)$ est continue, et donne une fonction $F(\nu)$ également continue. Il est à noter que $F(\nu) \in \mathbb{C}$ même si $f(t) \in \mathbb{R}$, comme dans la plupart des cas qui nous intéresse. On peut décomposer la fonction F en termes d'amplitude A et de phase ϕ.

$$F(v) = A(v)e^{i\phi(\nu)} \implies \begin{cases} A(\nu) = |F(\nu)| \\ \phi(\nu) = atan(\frac{Im(F(\nu))}{Re(F(\nu))}) \end{cases} \tag{3.3}$$

Cette décorrélation entre l'amplitude et la phase du domaine fréquentiel est très utile, surtout dans les cas où on veut une représentation invariante de phase.

La transformée de Fourier s'applique aussi dans le cas d'une fonction discrète. On parle alors de transformée de Fourier discrète (*Discrete Fourier Transform* ou DFT).

Soit $\vec{x} = (x_0, x_1, \ldots, x_{N-1})$, une séquence temporelle de N données à un taux d'échantillonnage f_e, donc représentant une durée $T = \frac{N}{f_e}$. La DFT de $\vec{x}$ est définie par la formule

suivante :

$$F_k = \sum_{s=0}^{N-1} x_s e^{-i2\pi k \frac{s}{N}} \qquad k = 0, \ldots, N-1 \tag{3.4}$$

Son inverse est donnée par :

$$x_s = \sum_{k=0}^{N-1} F_k e^{i2\pi k \frac{s}{N}} \qquad s = 0, \ldots, N-1. \tag{3.5}$$

Si on connaît le taux d'échantillonnage f_e, on peut trouver les équivalents de s et k en terme de temps et de fréquence :

$$t_s = s\frac{1}{f_e}, \tag{3.6}$$

$$\nu_k = k\frac{f_e}{N} = \frac{k}{T}. \tag{3.7}$$

Il existe un algorithme rapide pour calculer les F_k. Cet algorithme est connu sous le nom de transformée de Fourier rapide (*Fast Fourier Transform* ou FFT). Cet algorithme utilise la stratégie de diviser pour régner (*divide and conquer*) pour optimiser le temps de calcul [17].

Il est important de noter que, selon l'éq. 3.7, le pas fréquentiel $\Delta\nu = \frac{1}{T}$ est inversement proportionnel à la durée de l'échantillon. Par conséquent, effectuer une FFT sur une fenêtre de temps plus grande nous donne une meilleure résolution fréquentielle. Par contre, puisque la fenêtre de temps est plus grande, on a moins de précision sur la position temporelle des fréquences. Il y a donc un compromis à faire entre la résolution fréquentielle et temporelle.

Pour représenter un signal au complet dans le domaine fréquentiel, on effectue des FFTs sur des petites fenêtres de temps (e.g. 50 ms). La matrice contenant les FFTs l'une à la suite de l'autre est appelé spectrogramme (figure 3.2).

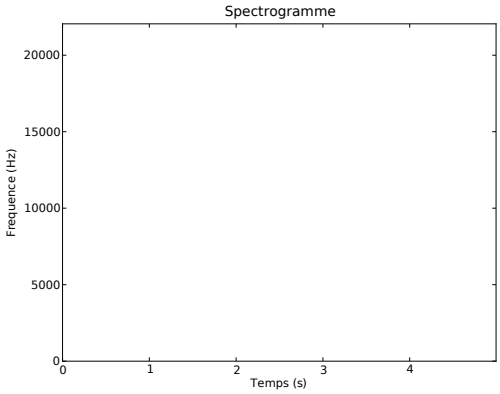


Figure 3.2 – Exemple de spectrogramme

3.2.3 Coefficients Cepstraux de Fréquence Mel

Les Coefficients Cepstraux de Fréquence Mel (MFCCs) ont été développés dans le domaine de l'analyse de parole et sont reconnus pour être de bonnes caractéristiques timbrales. Les MFCCs présentent plusieurs avantages. Premièrement, ils sont compacts, puisque l'on utilise typiquement entre 10 et 20 coefficients selon la tâche à résoudre. Les MFCCs ont comme propriété de décorréler la source et le filtre d'un son. Dans le domaine de la parole, cela se traduit par une séparation des caractéristiques de l'identité de l'interlocuteur, et celles des phonèmes prononcés.

Pour calculer les MFCCs, on passe le spectre à travers une banque de filtres triangulaires. La largeur de ces filtres varie en fonction de la fréquence selon une échelle logarithmique appelée échelle de Mel (*Mel scale*). Cette échelle est basée sur des études psycho-acoustiques de la perception humaine de la hauteur des sons. On calcule ensuite les premiers coefficients de la transformée cosinusidale discrète (Discrete cosine transform (DCT)) du logarithme des filtres. Pour plus de détails sur le calcul des MFCCs et sur la qualité de leur représentation du timbre musical, veuillez consulter [92].

Les MFCCs sont couramment utilisés dans plusieurs applications de MIR. Toutefois, comme on le verra au chapitre 7, ceux-ci n'offrent pas une représentation optimale pour des données musicales. La faible résolution fréquentielle des MFCCs ne permet pas de correctement représenter un demi-ton musical. De plus, les MFCCs n'offrent pas

une bonne représentation dans un contexte polyphonique où plusieurs sources se superposent.

3.3 Tâches de MIR

De plus en plus de musique est disponible en ligne. Avec la quantité de nouvelle musique qui apparaît chaque jour sur le web, il devient pratiquement impossible aux experts et aux amateurs d'écouter toute cette musique et de la décrire, la classer et de l'apprécier. Il est donc intéressant d'avoir des systèmes automatiques qui allègent la tâche des auditeurs leur permettant de filtrer la musique qui est pertinente pour eux.

Dans cette section, nous décrirons quelques tâches sur lesquelles le domaine du MIR s'est penché au cours des dernières décennies. Pour extraire des caractéristiques de bas niveau telles que la hauteur d'un son monophonique ou la pulsation, on a souvent recours à des algorithmes basés sur l'analyse de signal. Toutefois, pour extraire des concepts de plus haut niveau, tels que l'instrumentation ou le genre, il devient indispensable de faire appel à l'apprentissage machine.

3.3.1 Estimation de la hauteur du son

La détection de la hauteur du son (*pitch tracking*) cherche à estimer la fréquence fondamentale f_0 d'une note ou d'un son harmonique. Estimer f_0 est plus complexe qu'il n'y parait à première vue. On pourrait penser qu'il s'agit seulement de trouver le maximum du spectre. Toutefois, la hauteur perçue d'un son ne correspond pas nécessairement à la fréquence la plus saillante du spectre. Une note de musique est composée d'une série d'harmoniques multiples de f_0. Il se trouve que ces harmoniques sont très importantes au niveau perceptuel. Par exemple, on peut enlever complètement la fréquence f_0 d'une note, et on percevra quand même cette fréquence comme hauteur du son à cause de la disposition des harmoniques dans le spectre.

Dans le cas monophonique, c'est-à-dire lorsqu'une seule note est présente à la fois, plusieurs algorithmes relativement fiables nous permettent d'estimer f_0. Un des algorithmes les plus populaire, Yin [24], se base sur la fonction de différence du signal, une

fonction quelque peu similaire à l'auto-corrélation. Toutefois, il existe une multitude de techniques de détection de f_0. Quelques-unes de ces techniques sont comparées dans [39].

Dans le cas polyphonique, c'est-à-dire lorsque plusieurs notes sont superposées, le problème devient beaucoup plus complexe. Une grande partie du problème vient du fait que les harmoniques des différentes notes peuvent être superposées. En plus, le phénomène d'interférence (section 3.4.1) a pour conséquence que l'amplitude totale d'une fréquence donnée n'est pas égale à la somme des amplitudes de chacune des contributions. Le problème de détection de f_0 dans le cas polyphonique fortement lié au problème de transcription, qui tente de retrouver la partition à partir d'un extrait audio. Plusieurs méthodes ont été proposées pour attaquer ce problème [1, 84, 88].

3.3.2 Détection de pulsation rythmique

La détection de pulsation rythmique (*beat tracking*) cherche à trouver où se situent les pulsations principales d'un signal musical. En d'autres mots, trouver quand on devrait taper du pied en écoutant cette musique. Une difficulté de cette tâche est qu'il n'y a pas toujours de réponse claire. Si on demande à plusieurs personnes de taper du pied sur un même extrait musical, il se peut qu'on obtienne différentes réponses. Par exemple, il se peut que quelqu'un tape du pied deux fois plus vite qu'une autre, ou qu'une autre personne tape déphasée par rapport aux autres. Cela reflète le fait qu'un structure rythmique est souvent formée de différents niveaux hiérarchiques de pulsations, et que certains types de rythmes sous-entendent les pulsations principales (pensez au reggae ou au ska par exemple). Différentes approches pour détecter la pulsation sont présentées dans [19, 27, 57, 60].

3.3.3 Reconnaissance de genre

Pouvoir discriminer entre plusieurs genres musicaux (rock, disco, métal, jazz, classique, etc.) peut être utile pour effectuer des recherches ou pour classer une chanson ou un artiste dans une collection. Malheureusement, il n'est pas si facile de placer toute

la musique dans un certain nombre de boîtes. Un des problèmes avec la classification de genre, est qu'il n'y a pas de consensus précis sur la définition d'un genre. De plus, certains artistes influencés par différents genres créent de la musique qui emprunte des caractéristiques à plusieurs styles. Ou encore, une même chanson peut contenir plusieurs sections différentes appartenant à des genres différents.

On pourrait croire que l'oreille humaine est bonne pour distinguer les genres musicaux. Toutefois, Gjerdingen et al. (2008) [40] ont trouvé que les participants à une expérience avaient un taux de succès de seulement 70% quand on leur demandait de classer des extraits musicaux de 3 secondes entre 10 genres. Dans ces expériences, la vérité absolue venait de la classification donnée par les maisons de disques.

Un des ensembles de données souvent utilisé pour comparer les classifieurs de genre est l'ensemble de données de Tzanetakis [95]. Cet ensemble de données contient 1000 extraits de 30 secondes séparés en 10 genres.

La plupart des classifieurs de genre automatiques utilisent un apprentissage supervisé sur un ensemble de caractéristiques extraites de l'audio [7, 69, 95]. Certains modèles ont utilisé un encodage parcimonieux de l'audio dans le domaine temporel [76] et spectral [43]. Certaines autres approches utilisent des modèles séquentiels pour construire un représentation sémantique de la musique [21, 90]. Le meilleur résultat reporté sur l'ensemble de Tzanetakis est décrit dans [86] et utilise une technique de factorisation de tenseurs non-négatifs. Les défis et les motivations de la reconnaissance de genre automatique sont énoncés par McKay et al. (2006) [81]. Bergstra (2006) [9] présente une bonne description de l'état de la recherche en reconnaissance de genre.

3.3.4 Reconnaissance d'instrument

La reconnaissance d'instrument, comme son nom l'indique, consiste à identifier le ou les instruments présents dans un extrait audio. Souvent, on ne cherche pas à identifier un instrument en particulier, mais plutôt une classe d'instruments (pianos, instruments à vents, etc.). Toutefois, il existe une multitude de façons de classifier les instruments et les chercheurs ont de la difficulté à s'entendre sur une taxonomie commune.

Les premières recherches comparatives sur les qualités timbrales des instruments ont

été effectuées il y a plus de 30 ans [42, 80, 102]. Ces recherches psycho-acoustiques visaient à construire un espace de timbres dans lequel des instruments ayant des timbres similaires seraient proches. La topologie de l'espace ainsi obtenu a permis d'identifier les axes principaux de variation du timbre.

Beaucoup de recherche sur la reconnaissance d'instrument a été conduite sur des sons isolés et de l'audio monophonique [2, 33, 35, 38, 58, 77, 78, 98]. Un bonne revue de ces différentes approches est décrite dans [51]. Quelques travaux plus récents portent sur la reconnaissance d'instrument dans l'audio polyphonique et des mixtures d'instruments [29, 32, 36, 48, 49, 56, 67, 70, 71]. La polyphonie ajoute beaucoup de complexité au problème. Dans ce cas, les techniques de détection de hauteur de son et de séparation de source peuvent être utiles.

3.3.5 Étiquetage automatique

L'étiquetage automatique (*autotagging*) consiste à décider quels descripteurs, ou étiquettes, s'appliquent à un extrait musical, et lesquels ne s'appliquent pas. Plusieurs types de descripteurs peuvent s'appliquer. Par exemple, les mots peuvent décrire un genre (rock, hip-hop, blues, etc.), une instrumentation (guitare, piano, voix féminine, etc.), une émotion (joyeux, triste, agressif, etc.), un endroit géographique (Québécois, New York, Southern) ou tout autre concept pouvant être associé à un extrait musical. L'étiquetage automatique est donc une tâche plus générale que la reconnaissance d'instrument et de genre, puisqu'elle inclut celles-ci. Les étiquettes ainsi associées à un morceau de musique peuvent être utilisées pour mesurer la similarité de deux extraits, ou pour faire une recherche textuelle dans une base de données. L'état de la recherche en étiquetage automatique est décrite par Bertin-Mahieux et al. (2010) [12].

Certains services web tels que Last.FM mettent leur base de données d'étiquettes récoltées auprès de leurs utilisateurs à la disposition des chercheurs en MIR [11]. Également, Mandel et al. (2008) [72] ont construit une base de données intéressante à l'aide d'un jeu web où les joueurs gagnent des points en fonction de la qualité de leur étiquetage.

3.3.6 Similarité musicale

La tâche de similarité musicale cherche à établir une mesure de distance entre différentes oeuvres ou artistes. Le concept de similarité est difficile à définir en soi. Qu'est-ce qui fait que deux pièces soient similaires ? Est-ce que deux pièces d'un même artiste sont similaires ? Est-ce qu'une reprise d'une chanson par un autre groupe est similaire à l'originale ? Il n'est pas trivial de définir clairement ce qui constitue des pièces similaires, étant donné les différents aspects à considérer (instrumentation, harmonie, mélodie, rythme, paroles, voix, émotions, etc.). De plus, certains des aspects à considérer, tels que le propos des textes, sont hors de portée des systèmes d'analyse d'audio modernes.

Il est difficile d'obtenir une vérité absolue (*ground truth*) sur laquelle baser l'apprentissage supervisé d'un système de similarité musicale. Une méthode souvent utilisée pour obtenir un équivalent de vérité absolue est d'utiliser un ensemble de bibliothèques d'usagers. On suppose alors que, de manière générale, les oeuvres contenues dans la bibliothèque d'un usager donné seront similaires entre elles. Cela peut sembler être une supposition un peu forte, mais si on a accès a un grand nombre de bibliothèques d'usager, cela a tendance à fonctionner particulièrement bien. Un exemple d'ensemble de données qui fournit ce genre d'information est le *Million Song Dataset* [13].

3.3.7 Composition automatique

Plusieurs modèles de composition automatique existent déjà et réussissent, avec différents degrés de succès, à reproduire certaines caractéristiques musicales. Par exemple, l'algorithme de Mozer (1994) [83] peut reproduire des séquences simples et structurées localement, mais échoue à créer des compositions globalement cohérentes. L'algorithme de Eck (2002) [28] réussit à apprendre une structure d'accords de style blues et à improviser une mélodie à partir de ces accords. Celui de Paiement (2005) [85] est capable de composer une séquence d'accords de style jazz avec succès. Toutefois, malgré le fait que ces algorithmes soient capables de reproduire certaines caractéristiques musicales dans leurs compositions, ils échouent toujours le test de l'oreille humaine. Il est facile de distinguer ces compositions de compositions faites par un humain. Une des caractéristiques

de ces compositions est le manque structure globale et de cohérence à long terme.

On remarque que dans beaucoup de styles musicaux (pop, folk, blues, rock, etc.), les chansons sont constituées de plusieurs sections répétées plusieurs fois (e.g. couplets et refrains). Souvent, ces sections sont composées de phrases musicales répétées avec des variations. Donc, si on veut qu'un algorithme puisse apprendre correctement la structure globale de la musique, il est essentiel d'être capable de reconnaître des formes répétitives dans une chanson.

3.4 Défis reliés aux représentations d'audio musical

Le domaine de l'analyse de signal d'audio musical emprunte plusieurs techniques dérivées de domaines connexes tel que la vision ou la reconnaissance de parole. Toutefois, l'audio musical possède des caractéristiques différentes des images ou du langage parlé. Il est donc important de considérer les caractéristiques propres à la musique lorsqu'on tente d'adapter différentes techniques à l'analyse d'audio musical.

3.4.1 Interférence vs. occlusion

Soit un l'amplitude d'un signal audio $S(t)$. L'énergie instantanée E(t) du signal est proportionnelle au carré de l'amplitude :

$$E(t) \propto S(t)^2. \tag{3.8}$$

Soit deux signaux audio $S_1(t)$ et $S_2(t)$. Le signal $S_T(t)$ obtenu par la superposition de ces deux signaux est donnée par :

$$S_T(t) = S_1(t) + S_2(t). \tag{3.9}$$

Toutefois, en général, on a que :

$$E_T(t) \neq E_1(t) + E_2(t). \tag{3.10}$$

Ce phénomène, connu sous le nom d'interférence, se manifeste lorsque deux ondes interagissent ensemble. En audio, ce phénomène a pour conséquence que, lorsqu'il y a plusieurs sources sonores, de l'information est perdue par la superposition des sons. Ce phénomène est d'autant plus présent quand les sources sont en harmonie (i.e. elles émettent à des fréquences similaires, ou ayant des rapports entiers). Malheureusement (ou heureusement, selon le point de vue de l'auditeur), l'audio musical est très souvent constitué d'une multitude de sources en harmonie. Pour des tâches telles que la séparation de sources, la reconnaissance d'instruments ou la détection de hauteur de sons en contexte polyphonique, il est crucial de prendre en compte le phénomène d'interférence.

En vision, au contraire, lorsque deux objets opaques se superposent, l'un des deux objets sera caché en partie derrière l'autre objet. On dit alors qu'il y a occlusion. Bien sûr, certain objets peuvent aussi être transparents, auquel cas on doit considérer l'interaction entre deux objets. Dans un signal audio, il n'y a pas d'occlusion. Par analogie, on pourrait voir le phénomène d'interférence comme si chaque source sonore était *transparente*.

La différence entre occlusion et interférence est à considérer lorsqu'on passe du domaine de la vision à l'analyse d'audio. Par exemple, si on considère le sous-échantillonage (*pooling*) de caractéristiques, il se peut que différentes méthodes soient optimales pour des données avec occlusion ou non.

3.4.2 Monophonie vs. polyphonie

L'analyse du langage parlé s'effectue la plupart du temps en contexte monophonique. C'est-à-dire que, on peut considérer que, dans la plupart des conversations, il n'y a qu'un seul interlocuteur qui parle à la fois. La musique, au contraire, est souvent polyphonique. À part dans le cas simple dans lequel un seul instrument joue une seule note à la fois, il y a superposition de notes, d'instruments ou d'événement dans l'audio musical.

L'analyse d'audio polyphonique est évidemment plus complexe que l'analyse d'audio monophonique. Par conséquent, des représentations utiles pour analyser l'audio monophonique pourraient s'avérer déficientes lorsqu'on passe au polyphonique. Par exemple, les MFCCs, qui ont été optimisés pour la reconnaissance de phonèmes en contexte monophonique, pourraient être sous-optimaux dans un contexte musical po-

lyphonique.

3.4.3 Échellonage

La musique possède une structure temporelle particulière. Différents aspects de la musique (e.g. attaque, note, mesure, phrase, refrain) ont des échelles temporelles différentes. Chacun de ces aspects se produit à un ordre de longueur temporelle défini. Cela signifie que, globalement, la musique n'est pas invariante d'échelle. Au contraire, pour le traitement d'image naturelle, un objet dans une image peut varier de taille en fonction de sa distance à la caméra. Dans ce cas, on voudra utiliser des modèles invariants d'échelle pour bien représenter les données. Pour la musique, idéalement, on voudrait une représentation qui soit capable de représenter tous les aspects saillants à toutes les échelles temporelles. Dans ce cas, des modèles multi-échelles ou hiérarchiques pourraient être plus efficaces.

3.4.4 Modélisation séquentielle

Des algorithmes de modélisation séquentielle tels que le modèle de Markov caché (*Hidden Markov model* ou HMM) ou les N-grammes sont souvent utilisés pour modéliser le langage parlé. Ces modèles cherchent à évaluer la vraisemblance de séquences d'états possibles afin de pouvoir prédire le prochain état dans la séquence. C'est donc la structure locale qui est modélisée par ce genre d'algorithme.

Toutefois, ces modèles n'ont pas obtenu beaucoup de succès dans le domaine musical. Une des raisons pourquoi ces modèles échouent à modéliser la musique est que la musique comporte d'importantes dépendances à long terme. Par exemple, le prochain accord dans une séquence d'accords pourrait dépendre plus de l'accord douze mesures auparavant que du dernier accord. En effet les modèles cherchant à modéliser la structure globale à long terme de la musique ([28, 85]) ont tendance à avoir plus de succès que les modèles modélisant les dépendances temporelles locales.

CHAPITRE 4

PRÉSENTATION DES ARTICLES

Les 4 chapitres qui suivent présentent des travaux publiés dans le cadre des mes recherches. Ces travaux visent à optimiser les représentations de l'audio musical en combinaison avec différents modèles d'apprentissage machine. Les publications sont présentées en ordre chronologique. Les chapitres 5 et 6 démontrent les bénéfices de l'utilisation de l'apprentissage profond pour différentes tâches de MIR. Les chapitres 7 et 8 argumentent en faveur de représentations hiérarchiques à multiples échelles temporelles.

Ces travaux ont tous été publiés en tant qu'actes de conférence pour ISMIR (International society for music information retrieval). La conférence ISMIR est une rencontre annuelle d'une multitude de chercheurs du domaine de la recherche d'information musicale. Elle regroupe, entre autres, des chercheurs en informatique, génie électrique, musicologie et de bibliotéconomie.

Ma contribution personnelle pour tous les travaux présentés ici inclut la mise au point des expériences, l'écriture de la majeure partie du code, l'exécution des expériences, l'écriture des articles et la présentation des travaux lors de conférences.

4.1 Chapitre 5 : Automatic identification of instrument classes in polyphonic and poly-instrument audio

---

P. Hamel, S. Wood et D. Eck. Automatic identification of instrument classes in polyphonic and poly-instrument audio. In Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09), 2009.

4.1.1 Contexte

Cet article présente, à ma connaissance, la première application du DBN au domaine du MIR. On l'utilise ici dans le contexte de la reconnaissance d'instrument. Pour ce faire, nous avons dû générer notre propre base de données d'audio musical composée de

plusieurs centaines d'instruments échantillonnés ou synthétiques.

4.2 Chapitre 6 : Learning features from music audio with deep belief networks

P. Hamel et D. Eck. Learning features from music audio with deep belief networks. In Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR'10), 2010.

4.2.1 Contexte

Cet article présente une deuxième application du DBN au domaine du MIR. Dans cette publication, nous nous sommes penchés sur les tâches de reconnaissance de genre et d'étiquetage automatique. De plus, contrairement à l'article précédent, nous avons utilisé directement une représentation fréquentielle de l'audio afin de permettre l'apprentissage d'une meilleure représentation profonde

Dans cet article, nous avons également démontré qu'il est possible d'utiliser le transfert de l'apprentissage d'une tâche à l'autre grâce à une représentation profonde.

4.2.2 Impact

Cet article a soulevé un certain engouement envers l'apprentissage non-supervisé et l'apprentissage profond en MIR. Par exemple, dans [91], un DBN avec un structure similaire à celle que nous avons présenté a été appliqué au problème de reconnaissance d'émotion dans l'audio musical. Également, [26] utilise un réseau convolutionnel profond pour la reconnaissance de genre, d'artiste et de clé.

4.3 Chapitre 7 : Temporal pooling and multiscale learning for automatic annotation and ranking of music audio

P. Hamel, S. Lemieux, Y. Bengio et D. 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'11), 2011.

4.3.1 Contexte

Cet article apporte trois contributions qui améliorent l'état de l'art pour la tâche d'étiquetage automatique. Premièrement, nous avons démontré que les MFCCs, couramment utilisés dans le domaine, ne sont pas optimaux en tant que représentation de l'audio musical. Pour répondre à cette lacune, nous proposons une représentation spectrale moins compacte mais plus efficace. Deuxièmement, nous démontrons qu'il est préférable de combiner plusieurs fonctions de *pooling* lors de l'agrégation temporelle de représentations calculées sur de courtes fenêtres. Finalement, nous proposons un modèle d'apprentissage profond à différentes échelles temporelles qui permet d'améliorer encore plus la performance.

4.3.2 Impact

Un modèle construit à partir des résultats présentés dans cet article à été soumis au concours MIREX (Music Information Retrieval Evaluation Music Information Retrieval Evaluation eXchange) 2011 [44]. Ce modèle a obtenu la meilleure performance pour les deux tâches d'étiquetage automatique ainsi que pour trois des quatres tâches de classification d'audio musical. Cela démontre que les améliorations proposées dans l'article sont bel et bien applicables et efficaces dans un contexte pratique.

4.4 Chapitre 8 : Building musically-relevant audio features through multiple timescale representations

P. Hamel Y.Bengio et D. Eck. Building musically-relevant audio features through multiple timescale representations. In Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR'12), 2012.

4.4.1 Contexte

Cet article propose de combiner plusieurs représentations de l'audio calculées à différentes échelles temporelles afin d'obtenir une meilleure représentation des différents

aspects de l'audio musical. On y utilise la représentation proposée dans l'article précédent et nous proposons une généralisation du procédé d'extraction à différentes échelles. Grâce a cette représentation, nous repoussons encore une fois l'état de l'art pour la tâche d'étiquetage automatique.

CHAPITRE 5

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

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.

5.1 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 commercially 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 5.1.

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[5, 53]. 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 may be 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 [4].

The paper is organized as follows : In Section 5.2, we describe previous research in the domain of instrument recognition. In Section 5.3, we describe the system used to generate our audio database. In Section 5.4, we discuss the features extracted from the audio. In Section 5.5, we describe in detail the three classification models we employed. We then discuss our results in Section 5.6.

5.2 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

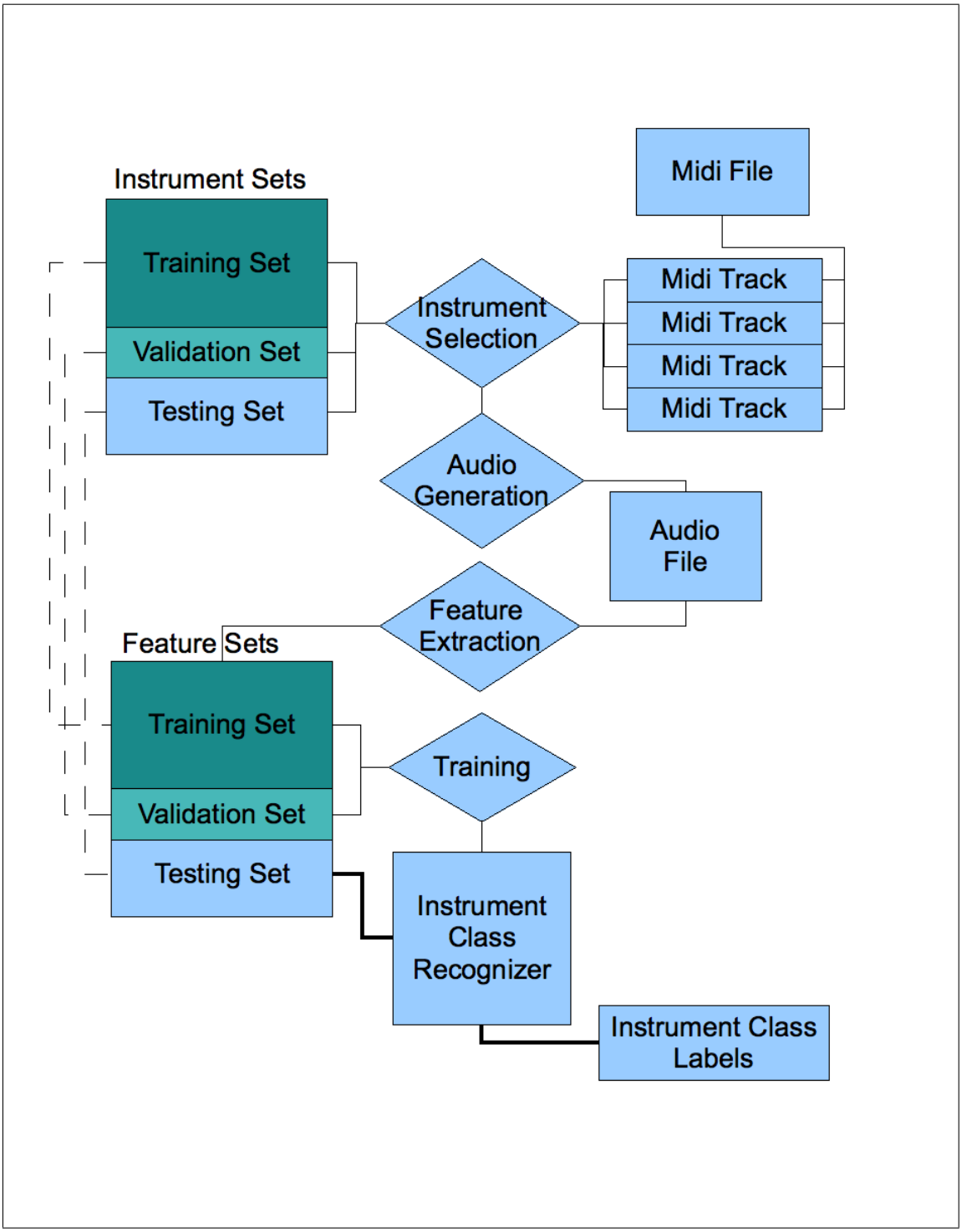


Figure 5.1 – Overview of our automatic instrument class recognizer model

similarity [80, 102]. 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 [2, 33, 35, 37, 38, 58, 77, 78, 98]. An overview of previous approaches to automatic instrument classification is described in [51]. Recent work deals with more complex and musically relevant sounds such as duets and polyphonies [29, 32, 56, 67, 70, 71]. There has also been much work done on the related task of predicting genre- and instrument-related tags from audio [11] [74] [93] 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 [29], missing feature theory and a Gaussian-mixture model (GMM) classifier was used to identify instruments in monophonic and polyphonic audio. In [71], 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 [33]. [56] presents a classification model using LDA and feature weighting using polyphonic audio and musical context information. In [68], instrument recognition in polyphonic audio is done by applying a post-process on a mid-level harmonic atoms representation. [70] 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 [48].

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.

5.3 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.

5.3.1 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.

5.3.2 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/`

5.4 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 [87].

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.

5.5 Models

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

5.5.1 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 [99].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 over-fitting, 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.

5.5.2 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 [20]. SVMs have been used for the task of instrument recognition with a good degree of success [70]. SVMs have the advantage of having fewer hyper-parameters to optimize than neural networks. We used cross-validation to optimize the hyper-parameters.

5.5.3 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 [4, 5]. 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)[53] or autoencoders[52, 100]. 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 [53]. 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 as 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.

5.6 Experiments

5.6.1 Experimental Setup

As in [70], 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

$$\text{Precision} = \frac{tp}{tp+fp}, \text{Recall} = \frac{tp}{tp+fn} \tag{5.1}$$

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

$$F = \frac{2(\text{precision} * \text{recall})}{\text{precision} + \text{recall}} \tag{5.2}$$

This can be simplified to

$$F = \frac{2tp}{2tp+fp+fn}. \tag{5.3}$$

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.

5.6.2 Results and Discussion

5.6.2.1 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 5.I. 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.

5.6.2.2 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 5.II.

We see that the DBN tends to perform better than both the SVM and the MLP in this

	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

Tableau 5.I – 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	

Tableau 5.II – 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

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.

5.6.2.3 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 5.III.

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 required 4 epochs of pre-training. The best MLP was

	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	

Tableau 5.III – 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

constructed with 60 hidden units.

5.6.3 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.

5.7 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 5.4, 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.

5.8 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.

CHAPITRE 6

LEARNING FEATURES FROM MUSIC AUDIO WITH DEEP BELIEF NETWORKS

Abstract

Feature extraction is a crucial part of many MIR tasks. In this work, we present a system that can automatically extract relevant features from audio for a given task. The feature extraction system consists of a Deep Belief Network (DBN) on Discrete Fourier Transforms (DFTs) of the audio. We then use the activations of the trained network as inputs for a non-linear Support Vector Machine (SVM) classifier. In particular, we learned the features to solve the task of genre recognition. The learned features perform significantly better than MFCCs. Moreover, we obtain a classification accuracy of 84.3% on the Tzanetakis dataset, which compares favorably against state-of-the-art genre classifiers using frame-based features. We also applied these same features to the task of auto-tagging. The autotaggers trained with our features performed better than those that were trained with timbral and temporal features.

6.1 Introduction

Many music information retrieval (MIR) tasks depend on the extraction of low-level acoustic features. These features are usually constructed using task-dependent signal processing techniques. There exist many potentially-useful features for working with music : spectral, timbral, temporal, harmonic, etc (see [87] and [9] for good reviews), and it is not always obvious which features will be relevant for a given MIR task. It would be useful to have a system that can automatically extract relevant features from the audio, without having to depend on *ad-hoc* domain-dependent signal processing strategies.

Among the most widely used frame-level features for audio-related MIR tasks Mel-Frequency Cepstral Coefficients (MFCCs). MFCCs take advantage of source/filter deconvolution from the cepstral transform and perceptually-realistic compression of spec-

tra from the Mel pitch scale. Because the first few MFCC values capture pitch-invariant timbral characteristics of the audio, they are commonly used in tasks where it is useful to generalize across pitch, such as multi-speaker speech recognition and musical timbre recognition.

Practically all audio-based music genre classification models use different types of acoustic features to drive supervised machine learning [7, 69, 74, 95]. These include sparse audio encodings in the time domain [76] and in the frequency (spectral) domain [43]. Other approaches use a Hidden Markov Model (HMM) to build a semantic representation of music [21, 90]. The best reported accuracy on the Tzanetakis dataset [95] for genre classification was achieved by a system that used auditory cortical representations of music recordings and sparse representation-based classifiers [86]. The challenges and motivations of genre classification are discussed in [81]. In these approaches it is difficult to know whether the acoustic features or the machine learning techniques are responsible for success. To address this we apply our model to the Tzanetakis dataset.

A closely related task to genre classification is that of "autotagging" (automatic tag-based annotation of music audio). As for genre classification, timbral and temporal features are often used to solve this task [12]. To test the robustness of our learned features, we applied them to the task of autotagging on the Majorminer dataset [73].

Some work in automatic feature extraction for genre classification have been done. In [82], automatic feature selection was done with genetic algorithms, and used for one-on-one genre classification. In our approach, we use a Deep Belief Network (DBN)[53] to learn a feature representation. DBNs have already been applied in some MIR tasks. In [47], a DBN is compared to other classifiers for the instrument recognition task. In [66], convolutional DBNs are used to learn features for speech recognition and for genre and artist classification.

Can we learn features for a given task directly from musical audio that would better represent the audio than engineered signal-processing features ? In this work, we investigate this question.

We propose a method to automatically extract a relevant set of features from musical audio. We will show that these learned features compare favorably against MFCCs and

other features extracted by signal-processing.

The paper is divided as follows. In Section 6.2, we describe the datasets that were used in our experiments. We then explain briefly the DBN model in Section 6.3. In Section 6.4 we describe the feature learning process. Then, in Section 6.5 we give the results of our features used in genre classification and autotagging tasks. Finally, we conclude and propose future work in Section 6.6.

6.2 Datasets

We used two different datasets in our experiments. The first one is the Tzanetakis' dataset for genre recognition. We trained our feature extractor over this dataset. To test the robustness of our learned features, we then applied these same features to the task of autotagging on the Majorminer dataset.

6.2.1 Tzanetakis

This dataset consists of 1000 30-second audio clips assigned to one of 10 musical genres. The dataset is balanced to have 100 clips for each genre. The dataset was introduced in [96], and have since been used as a reference for the genre recognition task.

6.2.2 Majorminer

This dataset for autotagging was introduced in [73]. The tags were collected by using a web-based "game with a purpose". Over 300 tags have been assigned to more than 2500 10 second audio clips. For our experiment, we used only the 25 most popular tags and compared our results to those obtained in [73].

6.3 Deep Belief Networks

In the last few years, a large amount of research has been conducted around deep learning[4]. The goal of deep learning is to learn more abstract representations of the

input data in a layer-wise fashion using unsupervised learning. These learned representations can be used as input for supervised learning in tasks such as classification and regression. Standard neural networks were intended to learn such deep representations. However, deep neural networks (i.e. networks having many hidden layers) are difficult or impossible to train using gradient descent[5]. The DBN circumvents this problem by performing a greedy layer-wise unsupervised pre-training phase. It has been shown [5, 53] that this unsupervised pre-training builds a representation from which it is possible to do successful supervised learning by "fine-tuning" the resulting weights using gradient descent learning. In other words, the unsupervised stage sets the weights of the network to be closer to a good solution than random initialization, thus avoiding local minima when using supervised gradient descent.

The Deep Belief Network (DBN) is a neural network constructed from many layers of Restricted Boltzmann Machines (RBMs)[5, 53]. A schematic representation is shown in Figure 6.1. A RBM is structured as two layers of neurons : a visible layer and a hidden layer. Each neuron is fully connected to the neurons of the other layer, but there is no connection between neurons of the same layer. The role of a RBM is to model the distribution of its input. We can stack many RBMs on top of each other by linking the hidden layer of one RBM to the visible layer of the next RBM. In our experiments, we used an algorithm inspired by Gibbs sampling called Contrastive Divergence (CD) to optimize our RBMs. Our focus here is on analyzing the performance of the DBN, not in explaining the technical details of DBNs. The main idea for our purposes is that that DBNs offer an unsupervised way to learn multi-layer probabilistic representations of data that are progressively "deeper" (nonlinear) with each successive layer. For technical and mathematical details see [5, 53]. We used the Theano[1] python library to build and train our DBNs.

[1] `http://deeplearning.net/software/theano/`

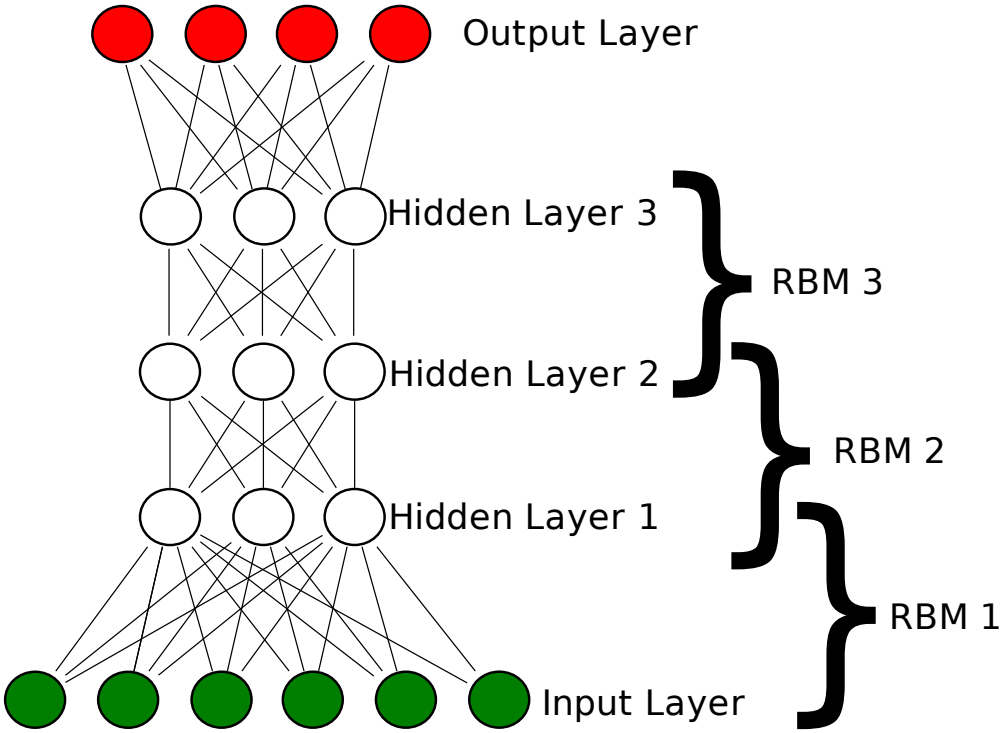


Figure 6.1 – Schematic representation of a DBN. The number of layer and the number of units on each layer in the schema are only examples. We do not require to have the same number of units on each hidden layer.

6.4 Learning the features

Our goal is is to learn a representation of audio that will help us to solve the subsequent tasks of genre classification and autotagging.

6.4.1 Training the DBN

To learn our representation, we split the Tzanetakis' dataset in the following way : 50% for training, 20% for validation and 30% for testing. We divided the audio into short frames of 46.44ms (1024 samples at 22050 Hz sampling rate). For each of these frames, we calculated the discrete Fourier transform (DFT). We kept only the absolute values of the DFTs, and considering the symmetry in the DFT, we ended up with inputs of dimension 513.

The DBNs were first pre-trained with the training set in a unsupervised manner. We then proceeded to the supervised fine-tuning using the same training set, and using the validation set to do early-stopping. The supervised step used gradient descent to learn a weighted mixture of activations in the deepest layer to predict one of 10 genre. Both soft max and cross-entropy costs were minimized with comparable results.

We tried approximately 200 different hyper-parameters combinations and chose the model with the best validation error on the frame level. The chosen DBN model is described in Table 6.I.

The classifier trained from the last layer of the DBN yields a prediction of the genre for each frame. We average over all predictions for a song and choose the highest score as the wining prediction. This gave us a prediction accuracy of 73.7%.

Once trained, we can use the activations of the DBN hidden units as a learned representation of the input audio. We analyzed the performance of each layer of the network independently, and also all the layers together. To illustrate what is learned by the DBN, in Figure 6.2 we have plotted a 2-dimensional projection of some of the representations used. The projection was done by using the t-SNE algorithm described in [97]. Notice how the clustering of the activations of the hidden layers is more definite than for the input or the MFCCs. As we will see in Section 6.5, this will improve the accuracy of the

Number of hidden layers	3
Units per layer	50
Unsupervised learning rate	0.001
Supervised learning rate	0.1
Number of unsupervised epochs	5
Number of supervised epochs	474
Total training time (hours)	104
Classification accuracy	0.737

Tableau 6.I – Hyper-parameters and training statistics of the chosen DBN

classifiers.

6.5 Classification using our learned features

In this section, we use our learned features as inputs for genre classification and autotagging. In the first task we explore different ways of using our features to get the best classification accuracy. In the second task, we use the method that gave us the best result in the genre recognition in order to do autotagging.

For both experiments, we use a non-linear Support Vector Machine (SVM) with a radial basis function kernel [20] as the classifier. It would also be possible to train our DBN directly to do classification. However our goal is to compare the DBN learned representation with other representations. By using a single classifier we are able to carry out direct comparisons.

6.5.1 Genre classification

6.5.1.1 Frame-level features

In our first experiment, we used our frame-level features as direct input to the SVM. Since the SVM doesn't scale well with large datasets, we subsampled the training set by randomly picking $10,000$ frames. We compared these accuracies to the accuracy of the SVM trained with MFCCs over these same frames of audio. As in Section 6.4.1, we used the frame predictions of a whole song and voted for the best genre in order to compute the test accuracy. The results for this experiments are shown in Table 6.II. We

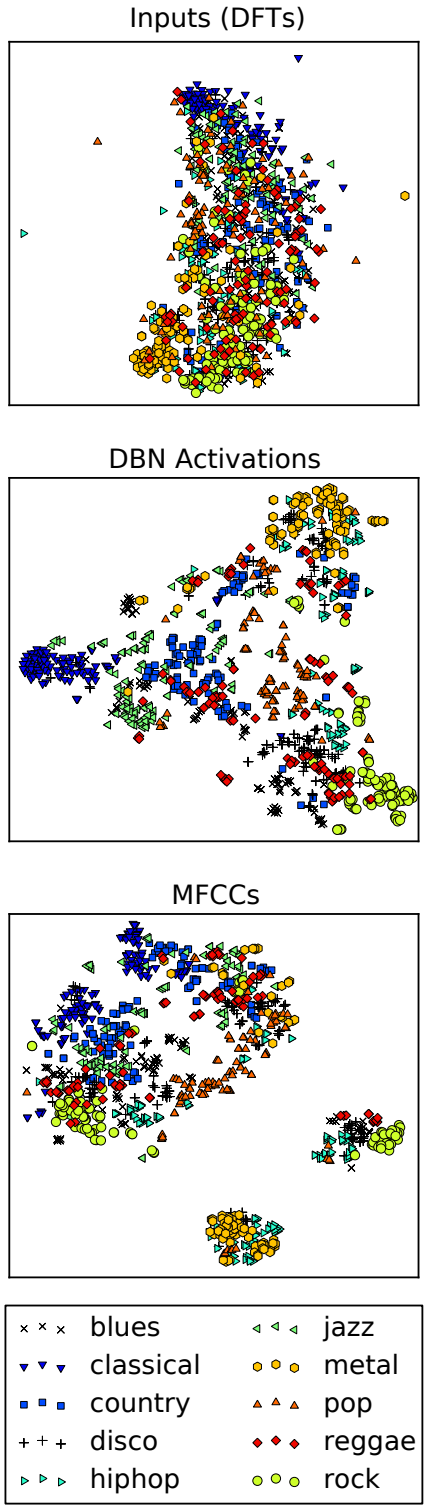


Figure 6.2 – 2-Dimensional projections of different representations of the audio with respect to their genre.

see that, at the frame level, our learned features performed significantly better than the MFCCs alone. We also see that the second layer seems to have the best representation out of the three layers. By using all the layers as the input, we don't see any improvement compared to the second layer alone. Since we used the same dataset here that we used for learning the features, we took care to reuse that same training, validation and testing splits as in Section 6.4, so as not to contaminate our testing set. Because our learned DBN representation was learned on a single test/train split, we were unable to do cross-validation on this dataset with the SVM classifier, since this would have given us a biased result.

6.5.1.2 Aggregated features

Bergstra et al [7] investigated the impact of feature aggregation on classification performance for genre recognition. It is demonstrated that aggregating frame-level features over a period of time increases classification accuracy. The optimal aggregation time depend depends on the nature of the features and the classifier, with many popular features having optimal aggregation times of between 3 and 5 seconds. With this in mind, we aggregated our features over 5 seconds periods. Thus, for each 5 seconds segment of audio (with 2.5 seconds overlap), we computed the mean and the variance of the feature vectors over time. This method not only raised our classification accuracy, but also reduced the number of training examples, thus accelerating the training of the SVMs. With the aggregation, our classification accuracy by jumped to 84.3%, which is better than the 83% accuracy reported in [7]. However, since this result was reported on a 5-fold cross-validation on the dataset, we cannot directly compare our results. More

	Accuracy
MFCCs	0.630
Layer 1	0.735
Layer 2	0.770
Layer 3	0.735
All Layers	0.770

Tableau 6.II – Classification accuracy for frame-level features

importantly we observe that our results are in general competitive with the state-of-the-art signal-processing feature extraction for the genre classification task. Also, given a fixed classifier (the nonlinear SVM) our learned representation outperforms MFCCs. As in Section 6.5.1.1, we see that the second layer gives the best representation of all the layers, but we gain a bit of accuracy by using all of the layers.

6.5.2 Autotagging

To test the robustness of our learned features, we tested their performance on an autotagging task. Following the results in Section 6.5.1, we used the activations of all the layers of the DBN aggregated on 5 second windows as inputs for the SVMs. We will refer to this set of feature as the DBN feature set. We compare it to a set of timbral and temporal features presented in [73]. We will refer to this set of feature as the MIM feature set. We used the same method as in [72] to train the SVMs over the dataset. The results for the 25 most popular tags in the dataset are shown in Figure 6.3 and summarized in Table 6.IV.

The results show that our features give a better classification performance for almost all the tags. In particular, our features performed significantly better better for tags such as 'rock', 'guitar', 'pop' and '80s'. Except for 'guitar', these particular tags represent genres, which is what our features were optimized to classify.

	Accuracy
MFCCs	0.790
Layer 1	0.800
Layer 2	0.837
Layer 3	0.830
All Layers	**0.843**

Tableau 6.III – Classification accuracy for features aggregated over 5 seconds

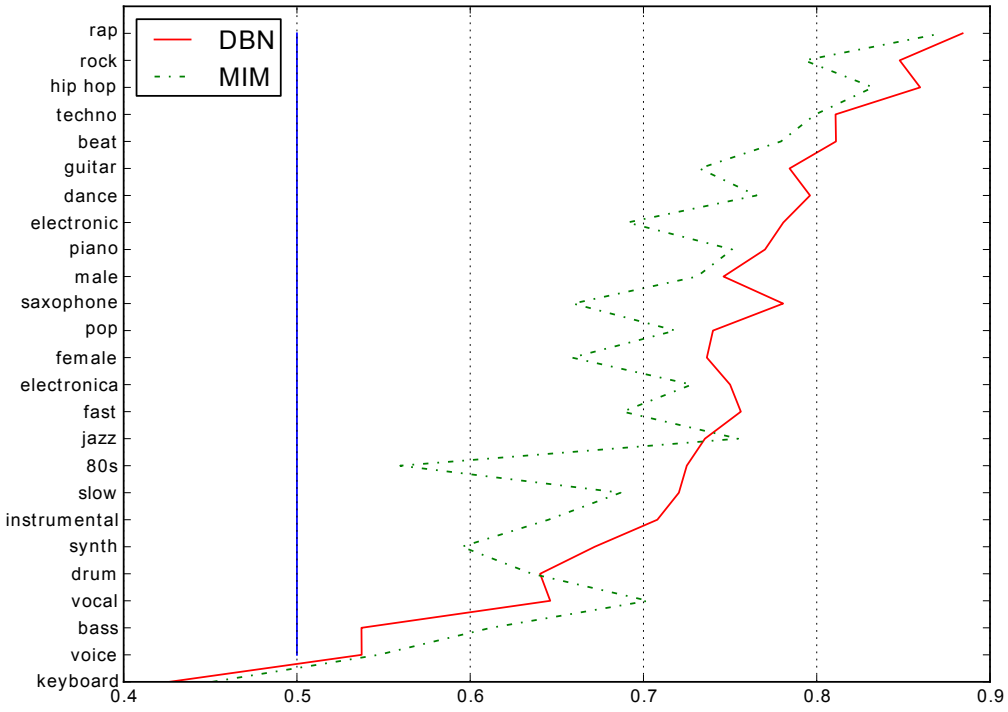


Figure 6.3 – Accuracy of the DBN and the MIM feature sets for the 25 most popular tags. As each tag training set was balanced for positive and negative examples, the vertical line at 0.5 indicates chance accuracy.

	Mean Accuracy	Standard Error
DBN	0.73	0.02
MIM	0.70	0.02

Tableau 6.IV – Mean and standard error of the autotagging results.

6.5.3 Discussion

From the results presented in Section 6.5.1 and Section 6.5.2, we see that it is indeed possible to learn features from audio relevant to a particular task. In the case of genre classification, our DBN features performed as well if not better than most signal-processing feature extraction approaches. The features were optimized to discriminate between the 10 genres shown in Figure 6.2, but we showed that these features were also relevant to describe many other tags, such as 'guitar', that were not related to genre. We believe this is evidence that a DBN can in fact learn to extract important and robust characteristics from audio. Another positive point is that, once the DBN is trained, the feature extraction from audio is very fast and can be done easily in real-time, which could be useful for many applications.

However, there are several areas for improvement. The main one is the long computation time necessary to train the DBN. The model that we used required a few days to train. This is mainly due to the size of the dataset. Since we used uncompressed audio frames overlapping over half a frame, the combination of the training and validation set required around 2 gigabytes of memory. There are many ways to reduce the size of the training set and to speed up the training. We could compress the DFTs with Principal Component Analysis (PCA). We could also aggregate the DFTs over small windows before sending them to the DBN. Randomly choosing a subset of the frames in the dataset could also help. Another solution would be to augment the mini-batch size to optimize the time of training process. However, it is not clear how each of these solutions will affect the quality of the representation. This requires further investigation.

Reducing the training time of a single model would also help to solve the second issue, which is the hyper-parameter search. As mentioned in Section 6.4.1, there are many hyper-parameters to optimize. It is not clear how the optimal hyper-parameters vary de-

pending on the input and the task. Current research on deep learning is investigating the matter, and some techniques to automatically adjust the hyper-parameters are being developed.

Another flaw of our model is that the features are extracted at the frame level only, so that our model cannot model long-term time dependencies. To better represent musical audio, we would need features that are able to capture the long-term time structure. Convolutional DBNs might provide a suitable model for time hierarchical representations [65].

6.6 Conclusion and Future Work

In this paper, we have investigated the ability for DBNs to learn higher level features from audio spectra. We showed that these learned features can outperform MFCCs and carefully-tailored feature sets for autotagging. These results motivate further research with deep learning applied to MIR tasks.

In future work, we will continue investigating ways to reduce the training time of our models. Furthermore, we will learn features over a wider range of datasets and MIR tasks. We are interested, for example, in using the unsupervised DBN training approach to observe a large amount of unlabeled audio data. Finally, we will continue to investigate how we can take advantage of structure found at multiple timescales in music. To this end, a hierarchical convolutional DBN may be appropriate.

6.7 Acknowledgements

Special thanks to Guillaume Desjardins and Michael Mandel for their contribution to the code used in the experiments. Thanks also to Simon Lemieux, Pierre-Antoine Manzagol and James Bergstra for helpful discussions, and to the Theano development team for building such useful tools. This research is supported by grants from the Quebec Fund for Research in Nature and Technology (FQRNT) as and the Natural Sciences and Engineering Research Council of Canada (NSERC).

CHAPITRE 7

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

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.

7.1 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 [12] 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 ap-

proaches that follow these steps and have been applied to the Magnatune dataset [61] in Section 7.3.1. 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 7.2. Then, we expose our experimental setup in Section 7.3. We present and discuss our results in Section 7.4. Finally, we conclude in Section 7.5.

7.2 Motivation

7.2.1 Choosing the right features

Choosing the right features is crucial for music classification. Many automatic annotation systems use features such as MFCCs[22, 59] 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[10]. Octave-based spectral contrast features [55] have been shown to also outperform MFCCs for genre classification. Thus, finding optimal features for audio

classification is still an open problem.

In section 7.3.3, we present a relatively simple audio preprocessing based on spectral energy bands and principal component analysis (PCA).

7.2.2 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 [16], 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 [103]. However, this method relies on onset detection methods which are not always reliable in all types of music.

7.2.3 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 [6, 63]. For instance, features learned with a Deep Belief Network over spectral amplitudes has been shown to outperform MFCCs for genre recognition and automatic annotation [45, 66].

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 [4] 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)[63] 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 7.3.5.2, we propose a hierachical model strongly inspired by CNNs.

7.3 Experimental setup

7.3.1 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 [62]. 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 [94], Mandel [72], Manzagol [75] and Zhi [22]. 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.

7.3.2 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.

7.3.3 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).

7.3.4 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.

7.3.5 Models

We used two different models in our experiments. The first one, described in Section 7.3.5.1, is a rather conventional system that applies feature extraction, pooling and clas-

sification in three separate steps. The second one, described in Section 7.3.5.2, applies learning both before and after the temporal pooling. The models are illustrated in Figure 7.1.

7.3.5.1 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 7.3.5.2. We will refer to this model as the Pooled Features Classifier (PFC) model.

7.3.5.2 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 7.3.4.

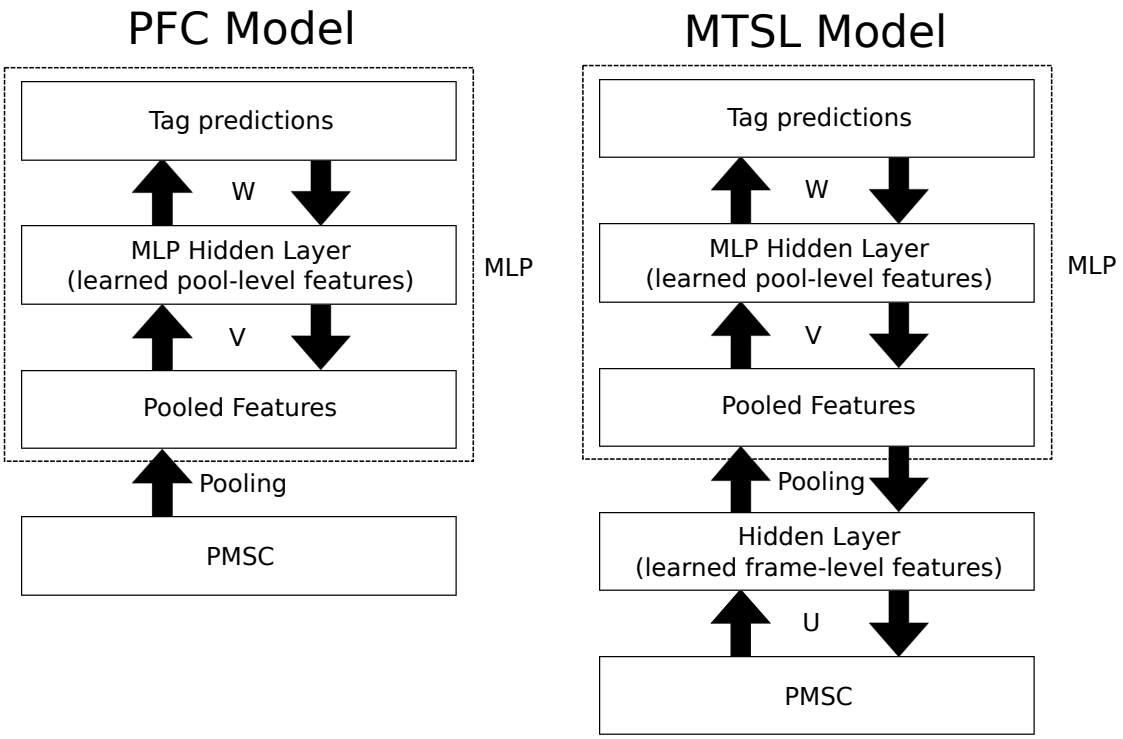


Figure 7.1 – Comparison of the PFC and the MTSL model. Upward arrows represent the flow of feed-forward information. Downward arrows illustrate the flow of the error back-propagation. U, V and W are weight matrices to be learned.

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.

7.4 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).

7.4.1 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 7.I. 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 [10] 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
MFCC(128)	0.767 +/- 0.004	9.3h
MFCC+PCA(128)	0.806 +/- 0.003	4.2h
Mel-spectrum(128)	0.853 +/- 0.008	5.2h
PMSC(120)	0.876 +/- 0.004	1.5h

Tableau 7.I – 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

7.4.2 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 7.2. 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.

7.4.3 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 7.3. 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, maximum and minimum gave the best performance.

7.4.4 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 7.4.2. The results for different sets of pooling functions is given in

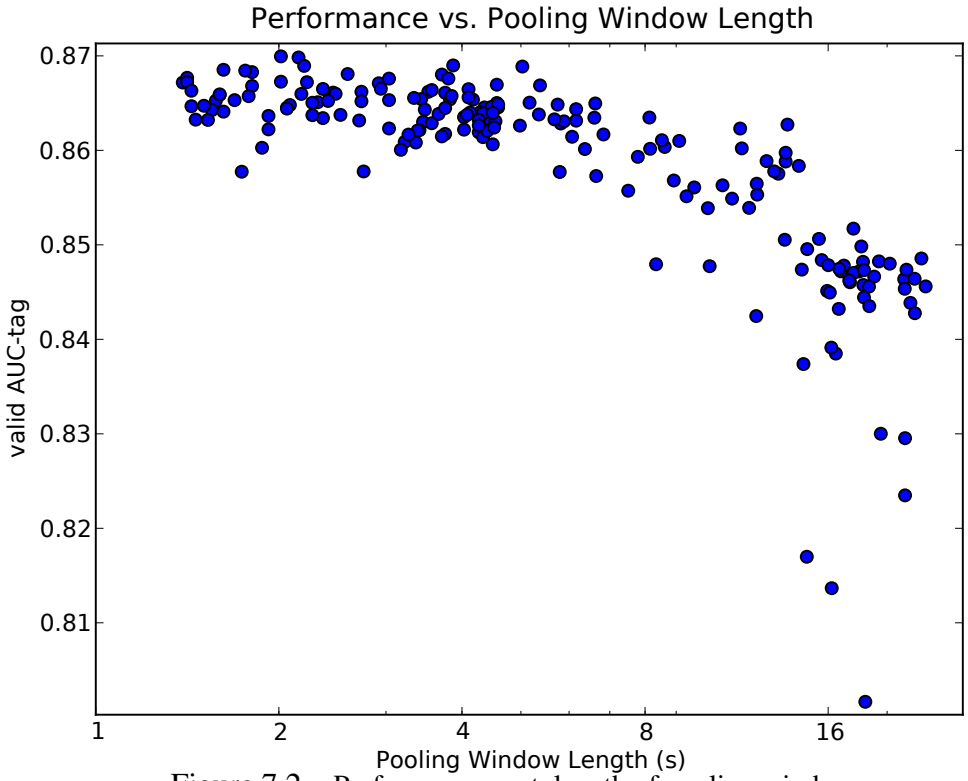


Figure 7.2 – Performance w.r.t. length of pooling window.

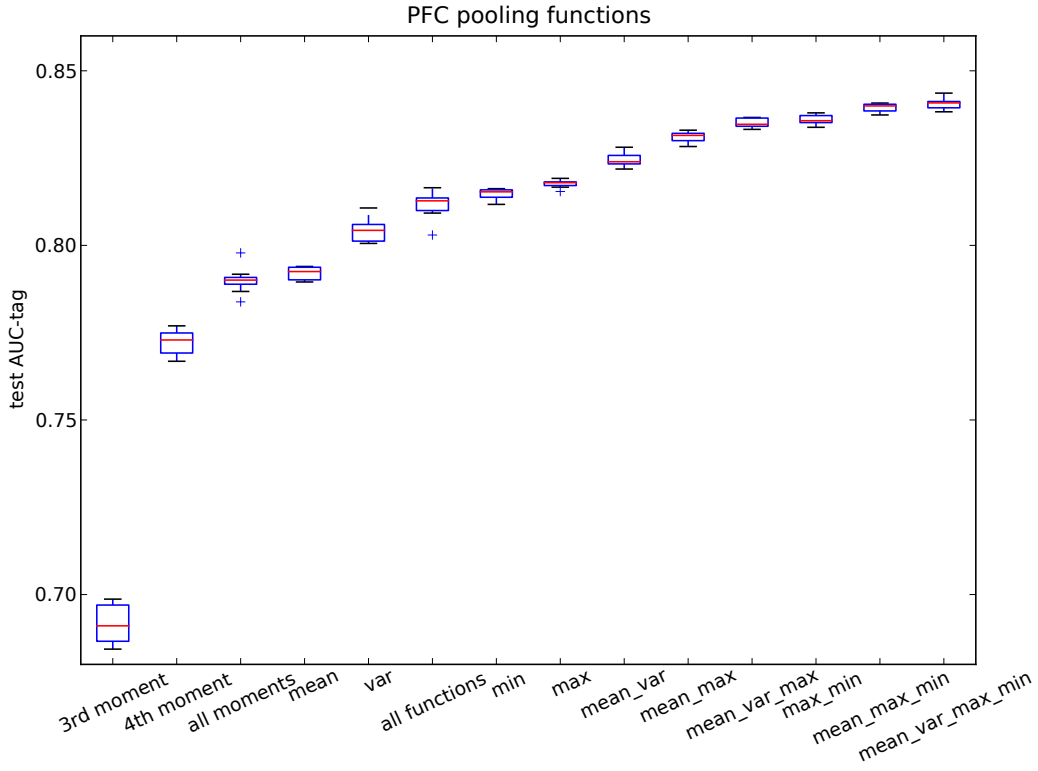


Figure 7.3 – Performance of different combinations of pooling functions for the PFC model

Figure 7.4. 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 7.4. Using more units further improves performance, but requires more computing time. The best MTSL models used around 350 units.

7.4.5 Comparative test performance

We compare the results of our models to those of the MIREX 2009 contest [1]. In Table 7.II, 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.

[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**

Tableau 7.II – 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.

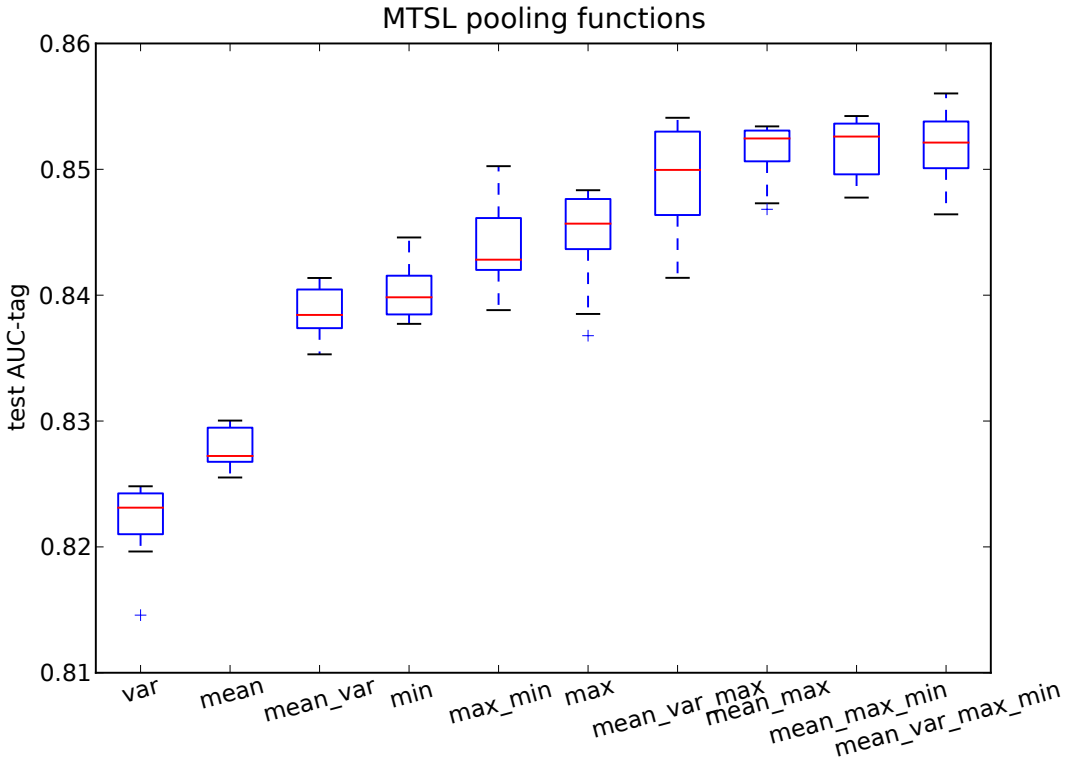


Figure 7.4 – Performance of different combinations of pooling functions for the MTSL model

7.5 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 [13] to test how well it scales to much larger music databases.

7.6 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 [8]. The authors would like to thank the Theano developper team.

CHAPITRE 8

BUILDING MUSICALLY-RELEVANT AUDIO FEATURES THROUGH MULTIPLE TIMESCALE REPRESENTATIONS

Abstract

Low-level aspects of music audio such as timbre, loudness and pitch, can be relatively well modelled by features extracted from short-time windows. Higher-level aspects such as melody, harmony, phrasing and rhythm, on the other hand, are salient only at larger timescales and require a better representation of time dynamics. For various music information retrieval tasks, one would benefit from modelling both low and high level aspects in a unified feature extraction framework. By combining adaptive features computed at different timescales, short-timescale events are put in context by detecting longer timescale features. In this paper, we describe a method to obtain such multi-scale features and evaluate its effectiveness for automatic tag annotation.

8.1 Introduction

Frame-level representations of music audio are omnipresent in the music information retrieval (MIR) field. Spectrograms, mel-frequency cepstral coefficients (MFCC), chromagrams and stabilized auditory images (SAI) are just a few examples of features that are typically computed over short frames. It has been shown that using frame-level features aggregated over time windows on the scale of a few seconds yields better results on various MIR tasks [7] than applying learning algorithms directly on frame-level features. However, the aggregation of frame-level features, also known as the bag-of-frames approach, does not model the temporal structure of the audio beyond the timescale of the frames. A simple method to get some information about short-time dynamics is to use the derivatives of the frame-level features. However, this method does not yield a representation that can model much longer temporal structure. Some alternative techniques to the bag-of-frames approach inspired by speech processing rely on the modelization

of the temporal structure with models such as HMMs [89]. A representation that could jointly model the short-term spectral structure and long-term temporal structure of music audio would certainly improve MIR systems.

In this paper, we take a step to improve the bag-of-frames approach by combining a set of features computed over different timescales. The idea is that longer timescale features, by modelling temporal structure, will give some context to the shorter timescale features which model spectral structure. The combination of multiple timescales could yield a general representation of the music audio that would be useful to solve various MIR tasks relying on audio features. In particular, we will show that a simple classifier trained over a multi-scale spectral representation of music audio obtains state-of-the-art performance on the task of automatic tag annotation. The multi-timescale representation that we introduce in this paper has the advantage of being a general purpose scalable method that requires no prior knowledge of the spectral or temporal structure of music audio.

The paper is divided as follows. First, in Section 8.2, we describe the current state of the research on multi-scale representations. Then, in Section 8.3, we describe our experimental setup. In Section 8.4 we discuss our results. Finally, we conclude in Section 8.5.

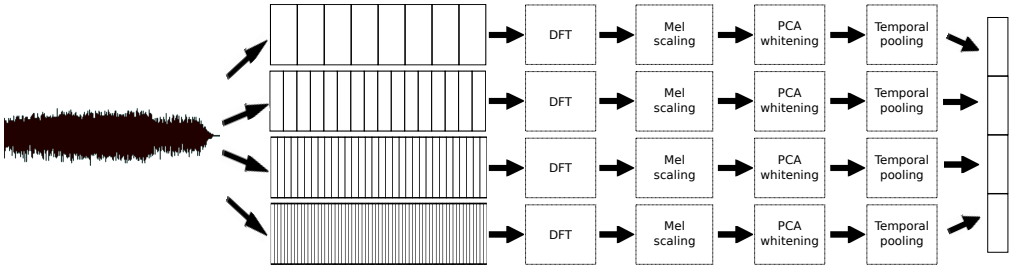


Figure 8.1 – PMSCs are computed in parallel at different timescales.

8.2 Multi-scale representations

Using representations at multiple scales allows much flexibility to model the structure of the data. Multi-scale representations offer a natural way to jointly model local and global aspects, without having prior knowledge about the local and global structures.

The idea of considering multiple scales is not new. It has been applied widely in the machine vision field. For example, pyramid representations[18] and convolutional networks [63] are just a few examples of multi-scale representations.

Recently, the MIR community as shown interest in taking advantage of multi-scale representations. Here are a few examples of recent work that has been done on multi-scale representation of music audio. In [79], structural change of harmonic, rhythmic and timbral features are computed at different timescales. This representation is used to build meaningful visualizations, although it has not been applied to music audio classification. In [34], boosting is applied on features at different timescales to optimize music classification. Although the validity of this method is demonstrated, it does not obtain state-of-the-art results on the CAL500 dataset. Learning features jointly at different timescales obtains state-of-the-art performance for automatic tag annotation [46]. However this model still depends on a bag of short timescale frames to build the long timescale representation, limiting the potential to model temporal dynamics. Deep convolutional networks have been applied to genre recognition in [66]. The authors show that classification performance for genre recognition and artist identification can be improved by using an unsupervised deep convolutional representation instead of raw MFCC features. Unfortunately, the results presented in this work are not comparable to other work in the field. In [3], scattering representations of MFCCs have been shown to improve music genre classification. The performance reported are comparable to other results reported on the same dataset. A bag-of-system approach have been proposed in [30] to combine models at various time resolutions.

8.3 Experimental setup

We used the TagATune dataset [62] in our experiments. TagATune is the largest dataset for music annotation available for research that provides audio files. It contains over 20,000 30-second audio clips sampled at 22050 Hz, and 160 tag categories. Our train, valid and test datasets contained 14660, 1629 and 6499 clips respectively.

We used the area under the ROC curve (AUC) averaged over tags (AUC-tag) as our main performance measure. We also use the AUC averaged over clips (AUC-clip) and precision at k for comparison with other models. For more details on these performance measures, see [46].

8.3.1 Multi-scale Principal Mel-Spectrum Components

In our experiments, we used Principal Mel-Spectrum Components (PMSCs) [46] as base features. PMSCs are general purpose spectral features for audio. They are obtained by computing the principal components of the mel-spectrum. PMSCs have shown great potential for the task of music tag annotation.

Moreover, it is quite simple to compute PMSCs at different timescales. The time length of the frame used to compute the discrete Fourier transform (DFT) determines the timescale of the features. To obtain multi-timescale features, we simply need to compute a set of PMSCs over frames of different lengths (Figure 8.1). The smallest DFT window we used was 1024 samples (46.4 ms). The size of the timescales grew exponentially in powers of 2 (1024, 2048, 4096, etc.).

We keep the same number of mel coefficients for all timescales. Thus, longer frames are more compressed by the mel-scaling, since the dimensionality of the output from the DFT is proportional to the frame's length. However, mel-scaling is more important for high frequency bins, while low-frequency bins are barely compressed by the mel-scaling. Fortunately, these high frequencies are already represented in shorter timescales where they are less compressed. In our experiments, we used 200 mel energy bands.

In our experiments, we found that using the log amplitude of the mel-spectrum yields better performance than using the amplitude.

PCA whitening is computed and applied independently on each timescale. In order to circumvent memory problems when computing the PCA, we limit the number of frame examples by randomly sub-sampling frames in the training set. We typically used around 75 000 frames to compute the PCA. It is also worth noting that we preserve all the principal components since we don't use PCA for dimensionality reduction, but rather to obtain a feature space with an approximate diagonal covariance matrix. The PCA whitening step decorrelates the features, which allows a more efficient temporal aggregation.

The principal components obtained for different timescales are shown in Figure 8.2. For each timescale, the first few principal components (those that account for the most variance in the data) tend to model global spectral shape. Subsequent components then model harmonic structure in the lower part of the mel-spectrum, and as we go up in the coefficients (and lower in the accounted variance), the components model structure in higher frequencies. It is interesting to notice the periodic structure in the components which shows how the harmonics are captured by the components. Also, if we compare components between timescales, we can observe that components tend to model a larger part of the mel-spectrum and exhibit more structure in the lower frequencies as we go higher in the frame size.

The next step consists of summarizing the features over a given time window by computing meaningful statistics. We refer to this step as temporal pooling. Following results from [46], we combined four pooling functions : mean, variance, maximum and minimum. These statistics are applied independently to each principal component through time and concatenated into a single feature vector for a given time window. In consequence, for each timescale we obtain a feature vector having four times the dimension of a single frame. Again, following results from [46], we fixed the pooling window at approximately 3 seconds for all experiments. Although, depending on how the frames were overlapped, this window length might vary for different timescales (see Section 8.4). The choice of the window length can be justified by the fact that 3 seconds would be enough for a human listener to label audio examples, but longer windows would give us less meaningful statistics for shorter timescales.

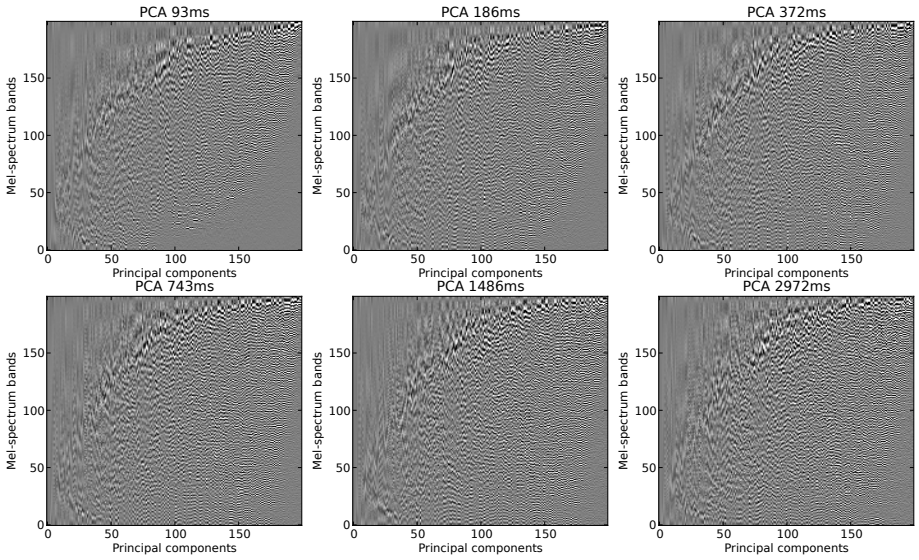


Figure 8.2 – PCA Whitening matrices for different timescales. The first few principal components tend to model global spectral shape. Subsequent components then model harmonic structure in the lower part of the mel-spectrum, and as we go up in the coefficients, the components model structure in higher frequencies.

By concatenating the pooled features from each timescale, we obtain multi-timescale PMSCs (Figure 8.1).

8.3.2 Multi-Layer Perceptron

The classifier we used is similar as the pooled feature classifier (PFC) model presented in [46]. However, in our case, the input pooled feature vector will tend to be larger, since it is obtained by concatenating many timescales.

We used a one-hidden layer artificial neural network, also known as multi-layer perceptron (MLP), as the classifier for all experiments. We kept the size of the network constant at 1000 hidden units for all experiments. The number of parameters (weights) in the system varies depending on the dimensionality of the input.

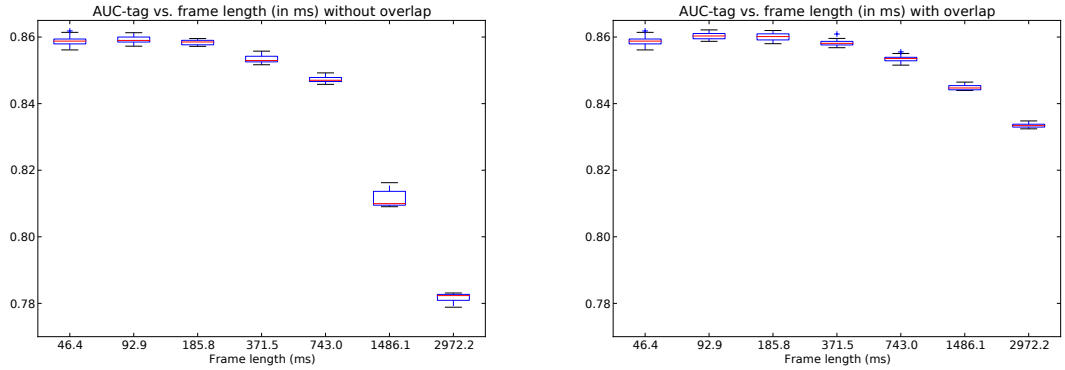
The input to the MLP is a multi-timescale PMSC representation a window of approximately 3 seconds of audio. In order to obtain tags for a full song in the test and validation phases, we simply average the MLP outputs over all windows from that song.

The MLP is well suited for multi-label classification like the music annotation task. The hidden layer acts as a latent representation that can model correlation between inputs as well as shared statistical structure between the conditional distributions associated with different targets (tags). This gives the MLP an advantage over other models such as the multi-class SVM, for which one would have to train a separate model for each tag. Also, the MLP scales sub-linearly in the number of examples, so it scales well to large datasets.

8.4 Results

In our experiments, we evaluated the performance of different timescales individually, and their combination for the task of automatic tag annotation.

In our first experiment, for a given timescale, we did not overlap frames. In consequence, longer timescales have fewer frame examples. In the extreme case, the longest timescale is the size of the pooling window, meaning that the max, mean and min are all equal, and variance is zero. Obviously, this is not ideal. As we can see in Figure

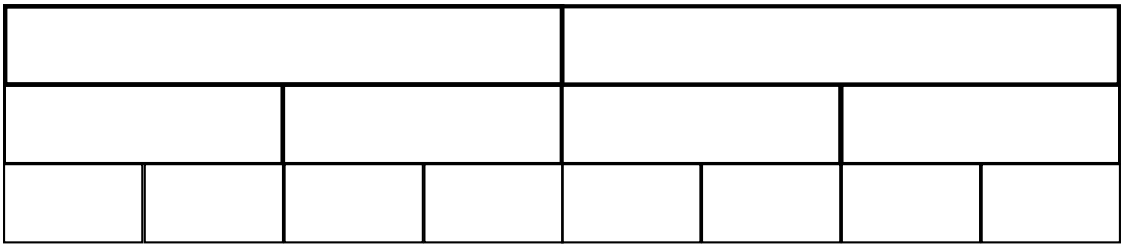


(a) (b)

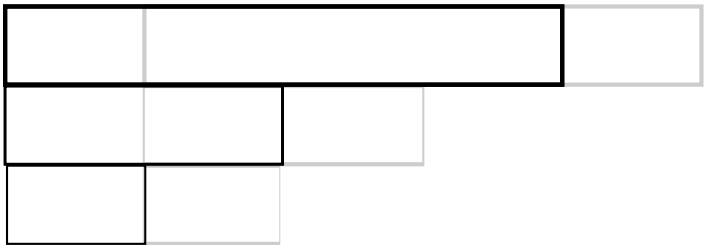
Figure 8.3 – AUC-tag for single timescale features without overlap (a) and with overlap (b). Shorter timescales tend to perform better than longer timescales, and performance generally improve when using overlapped frames.

8.3a, longer timescale perform worse than short timescales. However, we still see a significant advantage to using a combination of timescales. In Figure 8.5a, we show the performance of multi-timescale features. We combined timescales incrementally, starting from the shortest one to the longest one. For example, the representation with two timescales combines 46.4ms and 92.9ms frames, the one with three timescales combines 46.4ms, 92.9ms and 185.8ms frames, etc.

In order to obtain more examples for higher timescales, and yield more meaningful statistics for the temporal pooling, we considered using more overlapping between windows. In our second experiment, we used the same frame step for all timescales, corresponding to the smallest frame length, in this case, 46ms (Figure 8.4). We include all frames that start within the pooling window in the temporal pooling. This means that the longest timescale frames will overflow beyond the pooling window length up to almost twice the window length. Even though this method will give us the same number of frames to aggregate for each timescale, the longer timescales will still have much more redundancy than shorter timescales. Longer timescales perform significantly better with more overlap than without overlap, as we can see by comparing Figure 8.3a and 8.3b. The overlap also gives a boost of performance when combining timescales (Figure 8.5b).

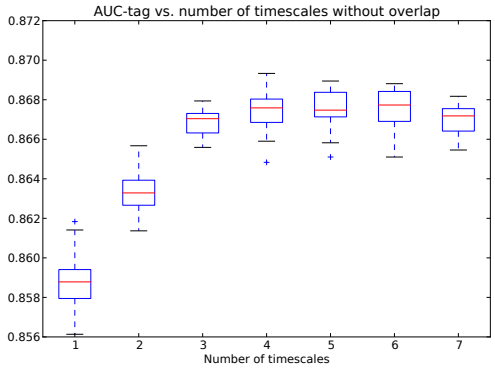


(a) no overlap

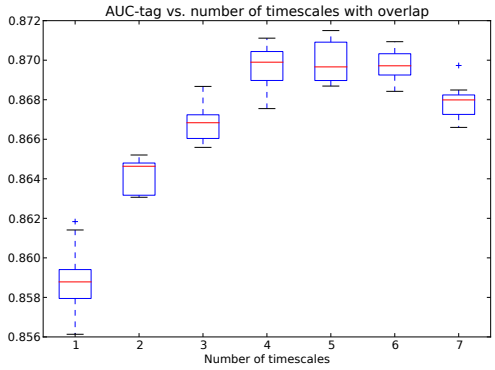


(b) overlap

Figure 8.4 – Illustration of frames without overlap (a) and with overlap (b).



(a)



(b)

Figure 8.5 – AUC-tag in function of number of timescales used without overlap (a) and with overlap (b). The combination of timescales always include the shorter timescales. For example, the representation with 2 timescales combines 46.4ms and 92.9ms frames, the one with 3 timescales combines 46.4ms, 92.9ms and 185.8ms frames, etc.

In Table 8.I, we show the test performance of the model that obtained the best AUC-tag on the validation set. We compare with two other state-of-the-art models : Multi-timescale learning model (MTSL) [46] and Music Understanding by Semantic Large Scale Embedding MUSLSE [104]. The multi-timescale PMSCs trained with the MLP obtains the best performance on all measures. Moreover, this model is a lot faster to train than the MTSL. For the TagATune dataset, the training time would typically be a few hours for the MLP compared to a few days for the MTSL.

8.5 Conclusion

Multi-timescale PMSCs are general purpose features that aim at jointly modelling aspects salient at multiple timescales. We showed that, for the task of automatic tag annotation, using multi-timescale features gives an important boost in performance compared to using features computed over a single timescale. Moreover, with a simple classifier, we obtain state-of-the-art performance on the TagATune dataset.

Multi-timescale PMSCs could potentially improve the performance of more complex learning models such as MTSL or MUSLSE. They could most likely be useful for other music information retrieval tasks such as genre recognition, instrument recognition or music similarity as well.

Although the timescales used in these experiments are not long enough to model many aspects of the temporal structure of music, the combination of multiple timescales of analysis allows to model some mid-level temporal dynamics that are useful for music classification. It is also a improvement on the typical bag-of-frames approach. Even

	Multi PMSCs	PMSCs	PMSCs + MTSL	MUSLSE
AUC-Tag	**0.870**	0.858	0.868	-
AUC-Clip	**0.949**	0.944	0.947	-
Precision at 3	**0.481**	0.467	0.470	0.476
Precision at 6	**0.339**	0.330	0.333	0.334
Precision at 9	**0.263**	0.257	0.260	0.259
Precision at 12	**0.216**	0.210	0.214	0.212
Precision at 15	**0.184**	0.179	0.182	0.181

Tableau 8.I – Performance of different automatic annotation models on the TagATune dataset

though we are still using frame level features, the concatenation of longer timescale representations puts short-time features in context.

In future work, it would be interesting to optimize the pooling window lengths independently for each timescale. This would allow longer timescale features to be aggregated over less redundant information and provide more relevant and stable statistics. It would also allow us to compute PMSCs over even larger timescales.

CHAPITRE 9

CONCLUSION

Cette thèse porte sur l'amélioration de représentations de l'audio musical en relation à l'apprentissage machine dans le cadre de la recherche d'information musicale. Aux chapitres 2 et 3, les notions de base de l'apprentissage machine et de la recherche d'information musicale ont été détaillées. Ensuite, au chapitre 4, les articles inclus dans cette thèse ont été introduits. Enfin, les chapitres 5 à 8 sont constitués de travaux publiés portant sur les représentations profondes et multi-échelles de l'audio musical.

9.1 Résultats importants

Les résultats des travaux présentés dans cette thèse nous permettent de tirer quelques conclusions importantes.

9.1.1 Domaine spectral vs. domaine cepstral

Les systèmes de MIR ont tendance à utiliser des caractéristiques trop compactes et mal adaptées au domaine musical. En particulier, les MFCCs 3.2.3 sont très répandus en tant que caractéristiques de l'audio.

On peut s'expliquer l'omniprésence des MFCCs par leur succès dans le domaine de reconnaissance de la parole. Ceux-ci permettent une séparation de la source et du filtre, ce qui est idéal pour isoler l'information relative aux phonèmes. Toutefois, dans un contexte polyphonique, tel que dans l'audio musical, cette caractéristique perd beaucoup de sa pertinence. De plus, les MFCCs n'utilisent qu'un nombre limité de bandes fréquentielles, limitant ainsi la résolution spectrale. Pour l'analyse de la parole, cela peut être souhaitable, puisque la hauteur absolue de la voix n'est pas pertinente au message, et les variation de hauteur dans les intonations sont larges. Contrairement à la parole, dans l'audio musical, la hauteur absolue des sons est critique. De plus, il est important d'utiliser une représentation ayant une résolution spectrale de l'ordre du demi-ton.

Les travaux présentés dans cette thèse démontrent que d'utiliser directement le spectre fréquentiel du signal audio permet de meilleures performances que les MFCCs. Les représentations spectrales sont moins compactes que les MFCCs, mais cela n'est pas un problème pour la plupart des algorithmes d'apprentissage machine. Nous avons même démontré que le MLP converge plus rapidement en utilisant une représentation spectrale comportant plus de 6 fois plus de dimensions que le MFCC (voir tableau 7.I)

On peut donc conclure que, pour des tâches de MIR dépendantes de l'audio musical, il est préférable d'utiliser une représentation spectrale qu'une représentation cepstrale tel que des MFCCs.

9.1.2 Apprentissage de représentations musicales

Dans chacun des travaux présentés dans cette thèse, nous avons appris une représentation à partir de l'audio. Dans les chapitres 7 et 8, nous avons utilisé l'une des formes les plus simple d'apprentissage non-supervisé, soit l'analyse en composante principales. De plus, au chapitre 7, nous avons présenté un modèle d'apprentissage supervisé à plusieurs échelles temporelles. Enfin, au chapitres 5 et 6, nous avons appris une représentation profonde de l'audio grâce au DBN.

Dans tous les cas, nous avons pu voir une amélioration de la performance comparé à d'autres méthodes ne comportant pas d'apprentissage de caractéristiques. De plus, au chapitre 6, nous avons démontré qu'il est possible de transférer l'apprentissage d'une tâche à une autre grâce à une représentation profonde. En particulier, la représentation profonde apprise pour la reconnaissance de genre performe mieux pour l'étiquetage automatique qu'un ensemble de caractéristiques obtenues par analyse de signal.

D'autres travaux récents démontrent également que l'apprentissage supervisé ou non-supervisé d'une représentation intermédiaire permet une meilleure performance pour plusieurs tâches de MIR [26, 50, 91, 104].

Donc, on peut en conclure qu'il est souhaitable d'intégrer une étape d'apprentissage de caractéristiques, supervisée ou non, pour obtenir un bon système de MIR.

9.1.3 Représentations profondes

Les chapitres 5 et 6 présentent différentes applications du DBN au MIR.

Au chapitre 5, nous avons comparé le DBN à deux modèles peu profonds pour la reconnaissance de classes d'instruments. Un des points importants à remarquer est que le DBN performe mieux que les autres modèles en particulier pour les cas où le nombre d'exemples positifs dans l'ensemble d'entraînement est faible (voir tables 5.II et 5.III). Cela pourrait être expliqué par le fait que la représentation profonde permet de mieux utiliser l'information apprise à partir d'exemples positifs des autres classes. Dans plusieurs problèmes de classification multi-classe ou multi-étiquette, il peut être difficile, voire même impossible de balancer les classes. Dans de tels cas, l'apprentissage d'une représentation profonde pourrait potentiellement aider à la classification des classes moins fréquentes.

Les résultats du chapitre 6, nous montrent qu'une représentation profonde permet d'obtenir de meilleurs résultats qu'avec des méthodes peu profondes. De plus on peut voir qu'il est avantageux d'entraîner un classifieur sur l'ensemble des couches d'un DBN, et non pas seulement à partir de la dernière couche, comme se fait typiquement l'ajustement fin. Finalement, une telle représentation reste assez générale pour pouvoir s'appliquer à un problème connexe.

Dans le chapitre 7, nous avons proposé un modèle d'apprentissage à différentes échelles temporelles. Ce modèle profond s'inspirant du réseau convolutionnel comporte une structure optimisée pour la tâche d'étiquetage automatique. On remarque que ce modèle obtient une meilleure performance que le modèle équivalent ne comportant pas d'étape d'apprentissage avant l'agrégation temporelle.

Nous avons ainsi démontré les bénéfices de l'apprentissage d'une représentation profonde pour diverses tâches de MIR.

9.1.4 Agrégation temporelle

La plupart des représentations de l'audio musical sont obtenues sur de très courtes fenêtres temporelles, typiquement de l'ordre de dizaines ou centaines de millisecondes.

Toutefois, pour plusieurs tâches de MIR, il a été démontré qu'il est souhaitable de résumer l'audio sur une période de l'ordre de quelques secondes.

Au chapitre 7, nous nous sommes penchés sur le problème de l'agrégation temporelle. Les résultats nous montrent qu'il est bénéfique d'effectuer une analyse en composantes principales sur la représentation spectrale avant l'agrégation. On remarque un nette amélioration de performance ainsi qu'une importante diminution du temps d'entraînement. La PCA capture les corrélations entre les bandes d'énergies spectrales, et redéfinit les bases de l'espace afin de minimiser la covariance entre ces nouvelles bases. Cela permet entre autres de conserver l'information relative aux harmonies lors de l'agrégation.

Un autre point que nous avons étudié est l'importance de la sélection de la fonction d'agrégation. Typiquement, les systèmes de MIR utilisent une combinaison de la moyenne et la variance des caractéristiques dans une fenêtre donnée. En général, dans le domaine des réseaux convolutionnels, on utilise soit le maximum, ou la moyenne pour résumer un représentation à une plus grande échelle. Dans nos travaux, nous avons trouvé que, pour la tâche de l'étiquetage automatique, la fonction qui performe le mieux, individuellement, est le maximum. Mais, plus important, nous avons montré qu'il est possible d'obtenir une meilleure performance en combinant plusieurs fonctions d'agrégation, soit le maximum, le minimum, la moyenne et la variance.

Ce résultat obtenu pour un tâche spécifique de MIR pourrait se généraliser à d'autres tâches de MIR, et même, on peut l'espérer, aux modèles convolutionnels en général. Dans notre cas spécifique, le rapport des échelles temporelles était important (de l'ordre de 64 fenêtres), alors que dans plusieurs applications des modèles convolutionnels, ce rapport est faible (typiquement 2, 3 ou 4). On peut supposer que la combinaison de fonctions d'agrégation soit souhaitable lorsque le rapport des échelles est important. Toutefois, ceci est un hypothèse qui reste à confirmer.

9.1.5 Représentation multi-échelles

Un des aspects souvent négligé, à tort, dans l'analyse d'audio musical, est l'aspect de la dynamique temporelle. L'agrégation temporelle de caractéristiques calculées sur de

courtes fenêtres ne nous permet pas d'obtenir une bonne représentation de la dynamique temporelle.

Afin de palier à cette lacune, nous avons proposé une représentation combinant plusieurs échelles temporelles au chapitre 8. Cette méthode a l'avantage d'être simple, rapide et échelonnable (*scalable*). Grâce à cette représentation, nous avons obtenu les meilleurs résultats jusqu'à ce jour pour l'étiquetage automatique.

9.2 Mot de la fin

Cette thèse a pour objectif d'apporter des améliorations aux méthodes utilisées pour analyser l'audio musical. Les résultats présentés ici montrent comment l'apprentissage de représentations profondes et multiéchelles permettent de meilleures performances pour plusieurs tâches de classification d'audio musical. On peut espérer que ces représentations soient utiles en général pour une multitude de tâches de MIR telles que la similarité musicale ou la génération automatique de listes d'écoute.

Certains résultats de ces recherches pourront potentiellement avoir un impact sur l'AM en dehors du domaine du MIR. Entre autres, les résultats sur les fonctions d'agrégations du chapitre 7 peuvent s'appliquer à d'autres modèles convolutionnels appliqués à d'autres types de données. Également, les représentations multiéchelles sont utiles pour plusieurs types de données présentant une structure spatiale ou temporelle. La méthode décrite au chapitre 8 pourrait inspirer de meilleures représentations dans le domaine de la vision, ou de la finance, par exemple.

Avec les progrès récents de l'AM et du MIR, les systèmes de MIR résolvent des tâches de plus en plus complexes. On commence maintenant à parler d'*intelligence musicale* plutôt que de recherche d'information. Dans un tel contexte, les représentations plus puissantes telles que les représentations profondes et multiéchelles sont d'autant plus importantes.

Il est difficile de prédire le futur de la production, distribution et consommation de la musique. Comme on l'a vu par le passé, le développement technologique joue un rôle crucial et entraîne des changements importants dans l'industrie musicale. Les progrès

technologiques émergeants du MIR auront certainement un impact sur la distribution et la consommation de la musique dans le futur. On peut espérer que ces changements permettront d'apporter des bénéfices autant pour les artistes, qui pourront distribuer leur musique plus facilement, que pour les auditeurs, qui auront un accès plus facile aux artistes pertinents grâce à des recommandations plus personnalisées.

BIBLIOGRAPHIE

[1] S. A. Abdallah et M. D. Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. Dans *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR'04)*, pages 10–14, 2004.

[2] G. Agostini, M. Longari et E. Pollastri. Musical instrument timbres classification with spectral features. *EURASIP J. Appl. Signal Process.*, 2003:5–14, 2003. ISSN 1110-8657.

[3] J. Andén et S. Mallat. Multiscale scattering for audio classification. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, pages 657–662, 2011.

[4] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009. ISSN 1935-8237.

[5] Y. Bengio, P. Lamblin, D. Popovici et H. Larochelle. Greedy layer-wise training of deep networks. Dans *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[6] Y. Bengio et Y. LeCun. Scaling Learning Algorithms towards AI. *Large-Scale Kernel Machines*, pages 1–41, 2007.

[7] J. Bergstra. Algorithms for Classifying Recorded Music by Genre. Thèse de maîtrise, Université de Montréal, 2006.

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

[9] J. Bergstra, N. Casagrande, D. Erhan, D. Eck et B. Kégl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65(2-3):473–484, 2006.

[10] J. Bergstra, M. Mandel et D. Eck. Scalable genre and tag prediction with spectral covariance. Dans *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR'10)*, 2010.

[11] T. Bertin-Mahieux, D. Eck, F. Maillet et 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.

[12] T. Bertin-Mahieux, D. Eck et M. Mandel. Automatic tagging of audio : The state-of-the-art. Dans Wenwu Wang, éditeur, *Machine Audition : Principles, Algorithms and Systems*. IGI Publishing, 2010.

[13] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman et P. Lamere. The million song dataset. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, 2011.

[14] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007. ISBN 0387310738.

[15] B.E. Boser, I.M. Guyon et V.N. Vapnik. A training algorithm for optimal margin classifiers. *Annual Workshop on Computational Learning Theory*, 1992.

[16] Y.-L. Boureau, J. Ponce et Y. Lecun. A theoretical analysis of feature pooling in visual recognition. Dans *International Conference on Machine Learning (ICML)*, 2010.

[17] G. Brassard et P. Bratley. *Fundamentals of Algorithmics*. Prentice-Hall, 1996.

[18] P. Burt et T. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Communications*, 9 :4(532–540), 1983.

[19] A.T. Cemgil et Bert Kappen. Monte Carlo methods for tempo tracking and rhythm quantization. *Journal of Artificial Intelligence Research*, 18(1):45–81, 2003.

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

[21] K. Chen, S. Gao, Y. Zhu et Q. Sun. Music Genres Classification using Text Categorization Method. *IEEE Workshop on Multimedia Signal Processing*, pages 221–224, octobre 2006.

[22] Shi-Huang Chen et Shih-Hao Chen. Content-based music genre classification using timbral feature vectors and support vector machine. Dans *Proc. Intl. Conf. on Interaction Sciences : Information Technology, Culture and Human*, ICIS. ACM, 2009. ISBN 978-1-60558-710-3.

[23] C. Cortes et V.N. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, septembre 1995. ISSN 0885-6125.

[24] A. de Cheveigné et Hideki K. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002. ISSN 00014966.

[25] G. Desjardins et Y. Bengio. Empirical evaluation of convolutional rbms for vision. Rapport technique, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 2008.

[26] S. Dieleman, P. Brakel et B. Schrauwen. Audio-based music classification with a pretrained convolutional network. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, 2011.

[27] D. Eck. Identifying Metrical and Temporal Structure with an Autocorrelation Phase Matrix. *Music Perception*, 24(2):167, 2006.

[28] D. Eck et J. Schmidhuber. Finding temporal structure in music : Blues improvisation with LSTM recurrent networks. Dans H. Bourlard, éditeur, *Neural Networks for Signal Processing XII, Proceedings of the 2002 IEEE Workshop*, pages 747–756, New York, 2002.

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

[30] K. Ellis, E. Coviello et G.R.G. Lanckriet. Semantic annotation and retrieval of music using a bag of systems representation. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, pages 723–728, 2011.

[31] D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent et S. Bengio. Why Does Unsupervised Pre-training Help Deep Learning ? *Journal of Machine Learning Research*, 11:625–660, 2010.

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

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

[34] R. Foucard, S. Essid, Lagrange M. et Richard G. Multi-scale temporal fusion by boosting for music classification. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, Miami, U.S.A, octobre 2011.

[35] A Fraser et I Fujinaga. Toward real-time recognition of acoustic musical instruments. Dans *Proceedings of the International Computer Music Conference*, pages 175–177, 1999.

[36] F. Fuhrmann, M. Haro et P. Herrera. Scalability, Generality and Temporal Aspects in Automatic Recognition of Predominant Musical Instruments in Polyphonic Music. Dans *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09)*, pages 321–326, 2009.

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

[38] I. Fujinaga et K. MacMillan. Realtime recognition of orchestral instruments. Dans *Proceedings of the International Computer Music Conference*, 2000.

[39] D. Gerhard. Pitch Extraction and Fundamental Frequency : History and Current Techniques, 2003.

[40] R. Gjerdingen et D. Perrott. Scanning the Dial : The Rapid Recognition of Music Genres. *Journal of New Music Research*, 37(2):93–100, 2008. ISSN 0929-8215.

[41] X. Glorot et Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. Dans *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 249–256, 2010.

[42] J.M. Grey. *An exploration of musical timbre*. Department of Music, Stanford University, 1975.

[43] R. Grosse, R. Raina, H. Kwong et A. Y. Ng. Shift-invariant sparse coding for audio classification. Dans *Proceedings of the Conference on Uncertainty in AI*, 2007.

[44] P. Hamel. Pooled features classification mirex 2011 submission. Music Information Retrieval Evaluation Music Information Retrieval Evaluation eXchange (MIREX), 2011.

[45] P. Hamel et D. Eck. Learning features from music audio with deep belief networks. Dans *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR'10)*, 2010.

[46] P. Hamel, S. Lemieux, Y. Bengio et D. Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, 2011.

[47] P. Hamel, S. Wood et D. Eck. Automatic identification of instrument classes in polyphonic and poly-instrument audio. Dans *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09) , Kobe, Japan*, 2009.

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

[49] T. Heittola, A. Klapuri et T. Virtanen. Musical instrument recognition in polyphonic audio using source-filter model. Dans *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09)*, pages 327–332, 2009.

[50] M. Henaff, K. Jarrett, K. Kavukcuoglu et Y. LeCun. Unsupervised learning of sparse features for scalable audio classification. Dans *Proceedings of International Symposium on Music Information Retrieval (ISMIR'11)*, 2011.

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

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

[53] G. E. Hinton, S. Osindero et Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[54] G.E. Hinton, T.J. Sejnowski et D.H. Ackley. Boltzmann Machines : Constraint satisfaction networks that learn., 1984.

[55] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao et L.-H. Cai. Music type classification by spectral contrast feature. Dans *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 113–116. IEEE, 2002. ISBN 0-7803-7304-9.

[56] T. Kitahara, M. Goto, K. Komatani, T. Ogata et 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. ISSN 1110-8657.

[57] A.P. Klapuri, A.J. Eronen et J.T. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14:342–355, 2006.

[58] A.G. Krishna et T.V. Sreenivas. Music instrument recognition : from isolated notes to solo phrases. Dans *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 04, pages 265–268, 2004.

[59] T. Langlois. A music classification method based on timbral features. Dans *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09)*, 2009.

[60] E.W. Large et J.F. Kolen. Resonance and the perception of Musical Meter. *Connection science*, 6(2):177–208, 1994.

[61] E. Law et L. von Ahn. Input-agreement : a new mechanism for collecting data using human computation games. Dans *Proceedings of the International Conference on Human factors in computing systems*, CHI. ACM, 2009. ISBN 978-1-60558-246-7.

[62] E. Law, K. West, M. Mandel, M. Bay et J. S. Downie. Evaluation of algorithms using games : the case of music tagging. Dans *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09)*, 2009.

[63] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard et L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1:541–551, December 1989. ISSN 0899-7667.

[64] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219.

[65] H. Lee, R. Grosse, R. Ranganath et A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Dans *26th International Conference on Machine Learning (ICML '09)*, pages 609–616. ACM, 2009. ISBN 978-1-60558-516-1.

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

[67] P. Leveau, D. Sodoyer et L. Daudet. Automatic instrument recognition in a polyphonic mixture using sparse representations. Dans *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, 2007.

[68] P. Leveau, E. Vincent, G. Richard et L. Daudet. Instrument-Specific Harmonic Atoms for Mid-Level Music Representation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):116–128, janvier 2008. ISSN 1558-7916.

[69] T. Li et G. Tzanetakis. Factors in automatic musical genre classification. Dans *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 143–146, 2003.

[70] D. Little et B. Pardo. Learning Musical Instruments From Mixtures Of Audio With Weak Labels. Dans *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, 2008.

[71] A. Livshin et X. Rodet. Musical instrument identification in continuous recordings. Dans *Proceedings of the 7th International Conference on Digital Audio Effects*, pages 222–226, 2004.

[72] M. I. Mandel et D. P. W. Ellis. Multiple-instance learning for music information retrieval. Dans *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, pages 577–582, 2008.

[73] M. I. Mandel et D. P. W. Ellis. A web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

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

[75] P.-A. Manzagol et S. Bengio. Mirex special tagatune evaluation submission. Music Information Retrieval Evaluation Music Information Retrieval Evaluation eXchange (MIREX), 2009.

[76] P.-A. Manzagol, T. Bertin-Mahieux et D. Eck. On the use of sparse time relative auditory codes for music. Dans *9th International Conference on Music Information Retrieval (ISMIR'08)*, pages 14–18, 2008.

[77] J. Marques et P J. Moreno. A study of musical instrument classification using Gaussian mixture models and support vector machines. Crl technical report series, Cambridge Research Laboratory, Cambridge, Mass, USA, 1999.

[78] K.D. Martin. *Sound-source recognition : A theory and computational model*. Thèse de doctorat, MIT, Cambridge, 1999.

[79] M. Mauch et M. Levy. Structural change on multiple time scales as a correlate of musical complexity. Dans *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR'11)*, 2011.

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

[81] C. McKay et I. Fujinaga. Musical genre classification : is it worth pursuing and how can it be. Dans *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, 2006.

[82] I. Mierswa et K. Morik. Automatic feature extraction for classifying audio data. *Machine Learning*, 58:127–149, 2005. ISSN 0885-6125.

[83] M.C. Mozer. Neural network composition by prediction : Exploring the benefits of psychophysical constraints and multiscale processing, 1994.

[84] E. Nichols et C. Raphael. Automatic transcription of music audio through continuous parameter tracking. Dans *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, 2007.

[85] J.-F. Paiement, D. Eck, S. Bengio et D. Barber. A graphical model for chord progressions embedded in a psychoacoustic space. Dans *ICML'05 : Proceedings of the 22nd international conference on Machine learning*, pages 641–648, New York, NY, USA, 2005. ISBN 1-59593-180-5.

[86] Y. Panagakis, C. Kotropoulos et G. R. Arce. Music Genre Classification Using Locality Preserving Non-Negative Tensor Factorization And Sparse Representations. Dans *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09)*, pages 249–254, 2009.

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

[88] S.A. Raczynski, N. Ono et S. Sagayama. Multipitch analysis with harmonic nonnegative matrix approximation. Dans *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, pages 38–386, 2007.

[89] J. Reed et C.-H. Lee. On the importance of modeling temporal information in music tag annotation. Dans *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009*, pages 1873–1876, 2009.

[90] J. Reed et C.H. Lee. A study on attribute-based taxonomy for music information retrieval. Dans *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, 2007.

[91] E. M. Schmidt et Y. E. Kim. earning emotion-based acoustic features with deep belief networks. Dans *Proceedings of the 2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, 2011.

[92] H. Terasawa, M. Slaney et J. Berger. Perceptual distance in timbre space. Dans *Proceedings of the International Conference on Auditory Display (ICAD05)*, pages 1–8, 2005.

[93] D Turnbull, L Barrington, D Torres et G Lanckriet. Semantic Annotation and Retrieval of Music and Sound Effects. *IEEE Transactions on Audio, Speech And Language Processing*, 16(2):467–476, 2008.

[94] G. Tzanetakis. Marsyas submissions to MIREX 2009. Dans *MIREX*, 2009.

[95] G. Tzanetakis et P. Cook. Musical genre classification of audio signals. *IEEE Trans. on Speech and Audio Processing*, 10(5):293–302, 2002.

[96] G. Tzanetakis, G. Essl et P. Cook. Automatic musical genre classification of audio signals. Dans *Proceedings of the 2nd International Conference on Music Information Retrieval (ISMIR 2001), Bloomington, Indiana*, 2001.

[97] L.J.P. van der Maaten et G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[98] E. Vincent et X. Rodet. Instrument identification in solo and ensemble music using independent subspace analysis. Dans *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04)*, 2004.

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

[100] P. Vincent, H. Larochelle, Y. Bengio et P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. Dans *ICML '08 : Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[101] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano et K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics Speech and Signal Processing*, 1989.

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

[103] K. West et S. Cox. Finding an optimal segmentation for audio genre classification. Dans *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR '05)*, 2005.

[104] J. Weston, S. Bengio et P. Hamel. Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval. *Journal of New Music Research*, 2011.