Université de Lille
Inria Lille-Nord Europe

École doctorale **ED Régionale SPI 72**

Unité de recherche **Équipe-projet MΘDAL**

Thèse présentée par **Maxime Baelde**

Soutenue le **20 septembre 2019**

En vue de l'obtention du grade de docteur de l'Université de Lille

Discipline **Mathématiques et leurs interactions**

Spécialité **Statistique**

Titre de la thèse

Modèles génératifs pour la classification et la séparation de sources sonores en temps-réel

Thèse dirigée par Christophe Biernacki

Composition du jury

Rapporteurs	Laurent Girin	professeur au Grenoble-INP	
	Stéphane Girard	directeur de recherche à l'Inria Grenoble Rhônes-Alpes	
Examinateur	Sophie Dabo	professeur à l'Université de Lille	président du jury
Invités	Raphaël Greff	directeur R&D à l'entreprise A-Volute	
	Pierre Chainais	professeur à l'École Centrale de Lille	
Directeur de thèse	Christophe Biernacki	professeur à l'Université de Lille détaché directeur de recherche à Inria	

L'Université de Lille n'entend donner aucune approbation ni improbation aux opinions émises dans les thèses : ces opinions devront être considérées comme propres à leurs auteurs.

Mots clés : temps-réel, classification audio, séparation de sources, apprentissage statistique, modèles génératifs

Keywords: real-time, audio classification, source separation, statistical learning, generative models

Cette thèse a été préparée dans les laboratoires suivants.

Équipe-projet MΘDAL
Inria Lille-Nord Europe
40 avenue du Halley
59650 Villeneuve d'Ascq
France

☎ (+33) 03 59 57 78 00
Site https://modal.lille.inria.fr/

Laboratoire Paul Painlevé
CNRS U.M.R 8524
59655 Villeneuve d'Ascq
France

☎ (+33) 03 20 43 48 50
Site https://math.univ-lille1.fr/

À ma femme Camille
et mon chat Calvin.

La phrase la plus excitante à entendre en science, celle qui annonce de nouvelles découvertes, n'est pas «Eureka !» mais «C'est drôle...»

Isaac Asimov

C'est à travers la science que nous prouvons, mais à travers l'intuition que nous découvrons.

Henri Poincaré

En science, on essaie de dire aux gens, de manière à être compris de tous, quelque chose que personne n'a jamais connu auparavant. Mais en poésie, c'est exactement le contraire.

Paul Dirac

Modèles génératifs pour la classification et la séparation de sources sonores en temps-réel

Résumé

Cette thèse s'inscrit dans le cadre de l'entreprise A-Volute, éditrice de logiciels d'amélioration d'expérience audio. Elle propose un radar qui transpose l'information sonore multi-canale en information visuelle en temps-réel. Ce radar, bien que pertinent, manque d'intelligence car il analyse uniquement le flux audio en terme d'énergie et non en termes de sources sonores distinctes. Le but de cette thèse est de développer des algorithmes de classification et de séparation de sources sonores en temps-réel. D'une part, la classification de sources sonores a pour but d'attribuer un label (son monophonique) ou plusieurs labels (son polyphonique) à un son. La méthode développée utilise un attribut spécifique, le spectre de puissance normalisé, utile à la fois dans le cas monophonique et polyphonique de par sa propriété d'additivité des sources sonores. Cette méthode utilise un modèle génératif qui permet de dériver une règle de décision basée sur une estimation non paramétrique. Le passage en temps-réel est réalisé grâce à un pré-traitement des prototypes avec une classification hiérarchique ascendante. Les résultats sont encourageants sur différentes bases de données (propriétaire et de comparaison), que ce soit en terme de précision ou de temps de calcul, notamment dans le cas polyphonique. D'autre part, la séparation de sources consiste à estimer les sources en terme de signal dans un mélange. Deux approches de séparation ont été considérées dans la thèse. La première considère les signaux à retrouver comme des données manquantes et à les estimer via un schéma génératif et une modélisation probabiliste. L'autre approche consiste, à partir d'exemples sonores présent dans une base de données, à calculer des transformations optimales de plusieurs exemples dont la combinaison tend vers le mélange observé. Les deux propositions sont complémentaires, avec chacune des avantages et inconvénients (rapidité de calcul pour la première, interprétabilité du résultat pour la deuxième). Les résultats expérimentaux semblent prometteurs et nous permettent d'envisager des perspectives de recherches intéressantes pour chacune des propositions.

Mots clés : temps-réel, classification audio, séparation de sources, apprentissage statistique, modèles génératifs

Abstract

This thesis is part of the A-Volute company, an audio enhancement softwares editor. It offers a radar that translates multi-channel audio information into visual information in real-time. This radar, although relevant, lacks intelligence because it only analyses the audio stream in terms of energy and not in terms of separate sound sources. The purpose of this thesis is to develop algorithms for classifying and separating sound sources in real time. On the one hand, audio source classification aims to assign a label (e.g. voice) to a monophonic (one label) or polyphonic (several labels) sound. The developed method uses a specific feature, the normalized power spectrum, which is useful in both monophonic and polyphonic cases due to its additive properties of the sound sources. This method uses a generative model that allows to derive a decision rule based on a non-parametric estimation. The real-time constraint is achieved by pre-processing the prototypes with a hierarchical clustering. The results are encouraging on different databases (owned and benchmark), both in terms of accuracy and computation time, especially in the polyphonic case. On the other hand, source separation consists in estimating the sources in terms of signal in a mixture. Two approaches to this purpose were considered in this thesis. The first considers the signals to be found as missing data and estimates them through a generative process and probabilistic modelling. The other approach consists, from sound examples present in a database, in computing optimal transformations of several examples whose combination tends towards the observed mixture. The two proposals are complementary, each having advantages and drawbacks (computation time for the first, interpretability of the result for the second). The experimental results seem promising and allow us to consider interesting research perspectives for each of the proposals.

Keywords: real-time, audio classification, source separation, statistical learning, generative models

Équipe-projet MΘDAL
Inria Lille-Nord Europe – 40 avenue du Halley – 59650 Villeneuve d'Ascq – France

Remerciements

J'aimerais d'abord remercier Christophe Biernacki et Raphaël Greff pour la supervision de cette thèse. Leurs expertises respectives en modélisation probabiliste et traitement de signal audio m'ont permis d'appréhender les défis complexes soulevés par cette thèse.

Je remercie Laurent Girin et Stéphane Girard d'avoir acceptés de rapporter cette thèse. Je remercie également Sophie Dabo d'avoir accepté d'être dans le jury au titre d'examinateur et Pierre Chainais en tant qu'invité.

J'aimerais également remercier A-Volute et l'ANRT pour le financement de cette thèse CIFRE.

Un merci spécial aux membres d'A-Volute pour leur soutien durant ces trois ans de thèse avec eux. L'équipe R&D : Raphaël, Damien, Antoine, Nathan et les nombreux stagiaires et étudiants qui ont travaillé sur ce projet (Neila, Aurélie, Juliette,...). Je voudrais remercier l'équipe MΘdal pour les super séminaires du mardi ainsi que les Modal's Day, particulièrement Maxime, Quentin et Yaroslav pour avoir partagé le même bureau, ainsi que Adrien et Anne-Lise en tant que co-doctorants CIFRE.

Je remercie mes amis : Yoshi, Samy, Toutou, Jeff, Flo et Mathilde, pour avoir passé de très bons apéritifs, diners et parties de jeux (Mario Kart !) ainsi que le sport ensemble.

Je remercie enfin ma famille pour son soutien durant toute ma scolarité, et particulièrement ma femme Camille pour son soutien intangible et inaliénable durant toutes ces années.

Notations

Liste des variables

Variables «traitement de signal»

$x^{\text{temp}}(t)$ le signal audio analogique au temps t

T_e la période d'échantillonnage (F_e la fréquence d'échantillonnage) et q le facteur de quantification du signal

$\tilde{\mathbf{x}}^{\text{temp}}$ le signal échantillonné, $\hat{x}^{\text{temp}}(t)$ le signal quantifié, $\mathbf{x}^{\text{temp}}$ le signal audio numérisé avec T le nombre d'échantillons

$x^{\text{temp}}_{[t-\Delta T,t]}$ le signal audio numérisé observé au temps t sur un période $[t-\Delta T, t]$ avec ΔT la durée d'observation

$\mathbf{x}^{\text{freq}}$ le spectre de taille B de $\mathbf{x}^{\text{temp}}$ issu de la transformée de Fourier

Δt la taille d'une frame et δt la taille du décalage

N le nombre de frames dans un signal

$\mathbf{w} \in \mathbb{R}^{\Delta t}$ la fenêtre de traitement

t, f et τ respectivement le temps, la fréquence et la frame d'un signal

Variables «apprentissage statistique»

$\mathbf{x} \in \mathbb{R}^d$ le vecteur d'attributs de dimension d d'un algorithme d'apprentissage

$\mathbf{y}$ le vecteur des sorties d'un algorithme d'apprentissage et $\widehat{\mathbf{y}}$ son estimation

$(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ l'ensemble d'apprentissage de taille n

$\mathbf{z}$ le vecteur de label de taille K (le nombre de classes) d'une méthode de classification

$\boldsymbol{\theta}$ le vecteur de paramètres d'un modèle paramétrique et $\widehat{\boldsymbol{\theta}}$ son estimation

$\mathbf{X}$, $\mathbf{Z}$ les versions aléatoires des variables $\mathbf{x}$ et $\mathbf{z}$

$\boldsymbol{\pi}$ les proportions, $(\boldsymbol{\mu}_g)_{g=1}^G$ les vecteurs moyennes et $(\boldsymbol{\Sigma}_g)_{g=1}^G$ les matrices de covariance d'un modèle de mélange gaussien

$\mathbf{m}$ un modèle probabiliste et ν le nombre de paramètres de ce modèle

$\mathbf{h}$ les neurones, $\mathbf{w}$ les paramètres, b le biais et d la dimension d'une couche cachée d'un réseau de neurones

L le nombre de couches cachées d'un réseau de neurones

$\mathbf{X}$ une matrice de taille n lignes et d colonnes et $\mathbf{X}_{ij}$ l'élément de la ligne i et de la colonne j de $\mathbf{X}$

$\mathbf{W} \in \mathbb{R}^{n\times k}$ et $\mathbf{H} \in \mathbb{R}^{k\times d}$ les bases et activations d'une NMF, avec k le nombre de composantes

Variables de la méthode RARE

ϕ la proportion de mélange des spectres de puissance normalisées

$\mathbf{x}^{(q)}$ le spectre quantifié au facteur q
$\mathbf{p}^{(q)}$ les paramètres d'une loi multinomiale associé au spectre quantifié $\mathbf{x}^{(q)}$

Variables de séparation de sources

J le nombre de sources et M le nombre de microphones
$\mathbf{s}$ une source sonore et $\mathbf{x}$ son image spatiale
$\mathbf{A}$ la matrice de mélange de sources
$\mathbf{T}$ le vecteur de transformation d'un spectre
$\overline{\mathbf{T}} = [\mathrm{Re}(\mathbf{T}); \mathrm{Im}(\mathbf{T})]$ la vectorisation d'un vecteur complexe

Liste des opérateurs

Opérateurs «traitement de signal

$\mathbf{f}_{\mathrm{ech}}^{(T_e)}(\cdot)$ l'opérateur d'échantillonnage à la période d'échantillonnage T_e
$f_{\mathrm{quanti}}^{(q)}(\cdot)$ l'opérateur de quantification au facteur de quantification q
$f_{\mathrm{TF}}(\cdot)$ et $f_{\mathrm{TFD}}(\cdot)$ les opérateurs de la Transformée de Fourier classique et discrète respectivement

Opérateurs «apprentissage statistiques»

$\mathcal{R}(\cdot)$ l'opérateur de relation entrée-sortie d'une méthode d'apprentissage
$\mathcal{L}(\mathbf{y}, \mathcal{R}(\mathbf{x}))$ la fonction de perte entre la sortie théorique $\mathbf{y}$ et la valeur prédite $\mathcal{R}(\mathbf{x})$
$\ell(\boldsymbol{\theta})$ la log-vraisemblance des paramètres $\boldsymbol{\theta}$ et $\ell_c(\boldsymbol{\theta})$ la log-vraisemblance des données complètes des paramètres $\boldsymbol{\theta}$
$K(\cdot,\cdot)$ le noyau d'une SVM
$\boldsymbol{\phi}(\cdot)$ la fontion de mapping d'une SVM

Opérateurs des méthodes RARE et RASE

$f_{\mathrm{frame}}^{(\Delta t)}(\cdot)$ l'opérateur de découpage en frame
$f_{\mathrm{norm}}(\cdot)$ l'opérateur de normalisation
$f(\cdot)$ l'opérateur de calcul des attributs de la méthode RARE
$K^{(q)}(\cdot) = \mathrm{Mult}_B\left(\cdot; \mathbf{p}^{(q)}, q\right)$ le noyau multinomial
$\mathbb{E}_{p(\cdot)}[\cdot]$ l'espérance mathématique sous $p(\cdot)$

Sommaire

Liste des tableaux

Table des figures

Chapitre 1

Introduction et motivation

En électronique grand public deux domaines sont particulièrement représentés : l'image et l'audio. Ce dernier permet de transmettre de nombreuses informations, dont une information spatiale. De plus en plus de contenus utilisent des formats dits «multicanaux» (qui contiennent deux canaux (stéréo) ou plus) par opposition à «monocanal» (un seul canal). Les techniques de spatialisation sonore sont présentes dans de nombreux secteurs d'activité : le cinéma, les jeux vidéo plus récemment la réalité virtuelle et augmentée, voire la musique. Un exemple d'utilisation de spatialisation sonore est le jeu de tir à la première personne (FPS). En utilisant un système audio multicanal (5.1 ou 7.1), le joueur est capable de localiser ses ennemis virtuels grâce au son. La principale difficulté réside dans l'acquisition d'un tel système multicanal. De plus une écoute uniquement stéréo (au casque) non binauralisée ne permet pas de retranscrire toute l'information spatiale (seulement les canaux gauche et droit).

Basée sur ce constat, la société A-Volute a développé un produit qui transforme l'information spatiale contenue dans un flux audio en information visuelle. A-Volute est un éditeur de logiciel audio mondialement connu pour son logiciel d'amélioration audio Nahimic, et également pour d'autres logiciels en marque blanche. Le but d'un logiciel d'amélioration audio est de perfectionner l'expérience audio d'un utilisateur en corrigeant certaines parties du signal audio suivant le contexte (augmentation des basses ou des aigus, ajout de réverbération, suppression du bruit, *etc.*). Comme ce logiciel est surtout utilisé par la communauté des «gamers» (les joueurs de jeux vidéo), le logiciel propose des améliorations pour les jeux vidéo (mise en avant des explosions ou des bruits de pas par exemple), mais également pour l'écoute de musique ou le visionnage de film (amélioration de la voix notamment). Il est indispensable pour les joueurs de FPS de localiser les ennemis : ainsi une transcription visuelle des indices sonores est pertinente.

La traduction d'indices acoustiques utiles à la localisation avait déjà été traitée par Collins et Taillon [71, 22]. Ils proposent de représenter les évènements sonores par des icônes représentant le type de son (tir d'armes à feu, bruit de pas, *etc.*) ainsi que la direction d'émission de ce son. Afin de mettre en œuvre ce procédé, les développeurs de jeux vidéo devraient inclure ce traitement directement dans le jeu, ce qui va à l'encontre de leur habitude. Greff et Pham [72] ont donc proposé une méthode utilisant uniquement une analyse du flux audio multicanal. Cette méthode retranscrit visuellement certains attributs sonores comme le niveau ou la position spatiale.

La méthode d'A-Volute est une amélioration de cette dernière approche, basée sur l'analyse présentée par Goodwin et Jot [35]. Les flux audios générés par une application (un jeu vidéo

par exemple) sont analysés en temps-réel pour en déduire l'activité sonore instantanée dans un nombre fini de directions discrétisant l'espace de manière homogène. L'analyse effectuée n'exploitant que des principes d'acoustiques et des techniques de traitement du signal relativement communes, les données résultantes, bien que pertinentes, s'avèrent très peu intuitives et nécessitent d'être analysées et interprétées par l'utilisateur. On retrouve ce concept dans le produit Sound Tracker d'A-Volute par exemple. Ce dernier est un radar qui s'affiche en jeu et qui traite le son issu du jeu vidéo afin de donner la direction depuis laquelle un son prédominant provient. Un exemple est donné en Figure 1.1.

Cette thèse s'inscrit dans ce contexte applicatif. L'objectif est de convertir l'information sonore en une information de plus haut niveau, en développant et mettant en œuvre une analyse que l'on qualifiera *d'intelligente*. Au lieu de simplement raisonner en termes de niveau sonore, il s'agit de raisonner en termes d'évènements ou de types de sources sonores, et de pouvoir en temps-réel identifier et localiser les sources sonores principales composant la scène audio. Il s'agit en quelque sorte de pouvoir aboutir à la fonctionnalité décrite par Collins et Taillon [71, 22] en exploitant uniquement l'information contenue dans les flux audio.

Les objectifs de la thèse peuvent être regroupés en deux thèmes (l'ordre n'ayant pas d'importance) :

— *Classification* : il s'agit de développer une méthode permettant l'identification des sources présentes dans la scène sonore. La classification a pour objectif de donner un label à un son qui représente la classe d'appartenance de ce son (coup de feu, voix, *etc.*).
— *Séparation* : lorsque plusieurs sources sonores sont présentes en même temps, il s'agit de les séparer afin de manipuler plus précisément la scène sonore. La séparation fait donc référence à l'extraction (au sens signal) des différentes sources dans un mélange.

De plus ces deux traitements doivent être réalisés en *temps-réel*, c'est-à-dire que peu d'information est disponible pour le traitement et ce dernier doit être «rapide» (une définition plus formelle du temps-réel est donnée au Chapitre 3). Un aspect non étudié dans la thèse est la *localisation* de sources sonores. En effet, le Sound Tracker fait déjà une localisation de sources à partir de principes acoustiques. Or des méthodes d'apprentissage statistique permettent également de faire de la localisation. Des détails concernant cette problématique seront donnés dans le Chapitre 5.

Dans cette thèse, ces différentes problématiques sont traitées sous l'angle de *l'apprentissage statistique*. On suppose donc que dans chaque cas, on dispose d'exemples qui vont servir à entrainer un système de classification et de séparation. Ce genre de méthode est dite *supervisée* au sens où elle utilise des informations *a priori* sur les éléments à traiter, à l'inverse d'une approche *non supervisée* qui traite les éléments sans informations *a priori*. Plus spécifiquement, nous allons utiliser un outil particulier, la *modélisation probabiliste* basée sur des *modèles génératifs*. Le problème à résoudre sera formalisé de sorte à poser des hypothèses probabilistes (distributions de probabilités) sur les données à traiter. Cette modélisation est pertinente car elle permet de modéliser l'incertitude que l'on peut avoir sur les données. De plus, sa formalisation mathématique est bien posée et met en lumière les différentes hypothèses sous-jacentes du processus de génération de données.

Ainsi, cette thèse traite de différentes problématiques relatives au traitement de signal audio (principalement la classification et la séparation) et est organisée de la manière suivante.

Dans le chapitre 1 nous avons dressé le contexte industriel et académique de la thèse ainsi que les principales thématiques qui seront abordées dans le reste du manuscrit.

FIGURE 1.1 – Le Sound Tracker en situation de jeu vidéo. Il est symbolisé par le radar à droite de l'image. Dans ce cas précis un son semble provenir de l'arrière-gauche du personnage.

Dans le chapitre 2 nous présentons les notions de traitement de signal et d'apprentissage statistique utiles à la compréhension de la thèse. Les méthodes présentées dans ce chapitre serviront de référence pour comparer nos méthodes (notamment l'apprentissage profond).

Dans le chapitre 3 nous présentons la classification de sources sonores ainsi que la méthode développée dans cette thèse pour *la classification de sources sonores en temps-réel*. Celle-ci est basée sur un modèle probabiliste du spectre de puissance normalisé, attribut utilisé spécifiquement pour cet objectif (notamment la classification polyphonique). Un modèle génératif permet de dériver une règle de décision pour la classification de plusieurs frames audio. Cette méthode couvre à la fois les cas monophonique et polyphonique et donne des résultats encourageants surtout pour la tâche de classification polyphonique.

Dans le chapitre 4 nous présentons la séparation de sources ainsi que les méthodes développées pour *la séparation de sources audio en temps-réel*. Celle-ci consiste en un modèle génératif des spectres complexes. Nous avons proposé deux méthodes pour la résolution de cette tâche : la première utilise le formalisme des données manquantes, et la seconde utilise une famille de déformations à appliquer aux spectres de l'ensemble d'apprentissage telles qu'on puisse retrouver le signal mélangé.

Dans le chapitre 5 nous présentons les conclusions sur la thèse ainsi que des perspectives sur la localisation. Il s'agit de localiser (spatialement) les sources acoustiques et non juste un évènement. En effet pour le moment, le Sound Tracker ne fait qu'analyser la scène d'un point de vue global. La localisation consiste ainsi à calculer une direction spatiale dans laquelle se trouve principalement une source sonore.

Différentes annexes composent la fin de ce manuscrit. La liste des descripteurs audio usuels est disponible dans l'Annexe A. Des éléments de calculs pour certaines propriétés et estimations sont disponibles dans l'Annexe B. Des éléments de code d'implémentation numérique des différentes méthodes sont disponibles dans l'Annexe D. En effet cette thèse a fait l'objet de programmation intensive des méthodes développées ainsi que des méthodes concurrentes car l'objectif de la société est d'avoir une méthode qui fonctionne en temps-réel. Des résultats

numériques supplémentaires concernant la séparation de sources sont présents en Annexe C. Les articles de conférences et de journaux publiés et/ou soumis sont regroupés dans l'Annexe E.

L'art et la science de poser des questions
est la source de toute connaissance.

Thomas Berger

Chapitre 2

Quelques éléments de traitement du signal et apprentissage statistique pour la classification et la séparation de sources sonores

Ce chapitre présente les bases du traitement de signal audio numérique en temps-réel ainsi que les concepts d'apprentissage statistique qui seront utiles tout au long de ce manuscrit. La partie 2.1 présente le traitement de signal audio, c'est-à-dire ce qu'est un signal audio et les moyens de le traiter (échantillonnage et quantification, Transformée de Fourier, *etc.*). La partie 2.2 traite du traitement de signal en temps-réel, qui est un point central de la thèse (fenêtrage, overlap-add, *etc.*). La partie 2.3 introduit la théorie de l'apprentissage statistique, notamment les méthodes utilisées en classification et séparation de sources. Finalement la partie 2.4 dresse une conclusion sur ces différentes notions théoriques.

2.1 Traitement de signal audio

La plupart des notions relatives au traitement signal audio sont disponibles dans l'oeuvre de Julius O. Smith III [103] [1].

2.1.1 Son et signal numérique

Le son est une onde acoustique se propageant dans un milieu de transmission (l'air par exemple), dont la vélocité dans l'air est de $c = 340\text{m.s}^{-1}$. Cette onde est caractérisée par deux variables : vélocité et pression. En utilisant l'équation de Navier-Stokes, l'équation d'onde correspondante pour la variable pression est :

$$\nabla^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0, \tag{2.1}$$

1. Ces ressources sont également disponibles en ligne : https://ccrma.stanford.edu/~jos/pasp/

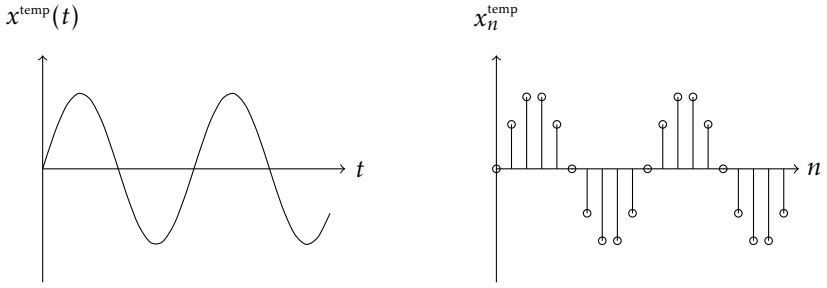


FIGURE 2.1 – Exemple de numérisation d'un signal analogique (ici un sinus pur). Les échantillons sont disponibles à des instants temporels discrétisés représentés par n et l'amplitude est quantifiée (elle ne peut prendre que certaines valeurs entières).

avec $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ l'opérateur de LAPLACE en 3D. Dans cette thèse on s'intéresse aux *signaux audio numériques audibles* qui sont des signaux audio numériques à bandes de fréquences limitées dans $[20\text{Hz}, 20\text{kHz}]$. Un signal numérique est une version *échantillonnée* et *quantifiée* d'un signal analogique (voir FIGURE 2.1). On note $x^{\text{temp}} : t \in \mathbb{R} \mapsto x^{\text{temp}}(t) \in \mathbb{R}$ un signal analogique. Le signal numérique correspondant est obtenu grâce aux étapes *d'échantillonnage* et de *quantification*.

Définition 1 (Échantillonnage)**.** *Durant l'échantillonnage, le temps t est discrétisé en utilisant la* période d'échantillonnage T_e. *Le signal échantillonné est défini par* $\tilde{\mathbf{x}}^{temp} = \mathbf{f}_{\text{ech}}^{(T_e)}(x^{temp}(t)) = (x^{temp}(nT_e))_{n\in\mathbb{Z}}$. *La* fréquence d'échantillonnage F_e *est définie comme l'inverse de la période d'échantillonnage* $F_e = 1/T_e$.

Définition 2 (Quantification)**.** *Durant la quantification, l'amplitude $x^{temp}(t)$ est discrétisée en utilisant un* facteur de quantification q *(arrondi ou troncature). Dans le cas d'une quantification utilisant l'arrondi, si $x^{temp}(t) \in [(k-1/2)\,q; (k+1/2)\,q]$, alors le signal quantifié est défini par $\hat{x}^{temp}(t) = f_{\text{quanti}}^{(q)}(x^{temp}(t)) = kq$.*

Le procédé de numérisation aboutit à un signal numérique noté $\mathbf{x}^{\text{temp}} = \left(x_n^{\text{temp}}\right)_{n=0}^{T-1} = \mathbf{f}_{\text{ech}}^{(T_e)} \circ f_{\text{quanti}}^{(q)}(x^{\text{temp}}(t))$ avec T la longueur du signal, qui est une version échantillonnée et quantifiée de $x^{\text{temp}}(t)$ (voir FIGURE 2.1). La fréquence d'échantillonnage F_e est une valeur importante car elle contrôle le niveau d'information de $x^{\text{temp}}(t)$ qui peut être capturé dans le signal numérique $\mathbf{x}^{\text{temp}}$. Le *théorème d'échantillonnage* de SHANNON donne une condition suffisante sur F_e pour garder toute l'information de $x^{\text{temp}}(t)$ lors de l'échantillonnage :

Théorème 1 (Théorème d'échantillonnage de SHANNON [98])**.** *Soit un signal $x^{temp}(t)$ qui ne contient pas de fréquences supérieures à $F_{\max}$. Une condition suffisante sur la fréquence d'échantillonnage pour que l'échantillonnage capture toute l'information contenue dans $x^{temp}(t)$ est $F_e \geq 2F_{\max}$.*

En traitement de signal numérique audio, cette condition est toujours vérifiée (en appliquant un filtre anti-repliement). En effet l'audition humaine se limite à des fréquences allant de 20Hz à 20kHz, et les fréquences d'échantillonnage usuelles sont $F_e = 44,1\text{kHz}$ (pour les formats type CD), et $F_e = 96\text{kHz}$ ou 192kHz (pour de la numérisation haute définition).

L'étude des signaux passe le plus souvent par une représentation spectrale, dans le domaine fréquentiel, du signal. Une représentation particulière, la transformée de FOURIER, est présentée dans la prochaine sous-partie.

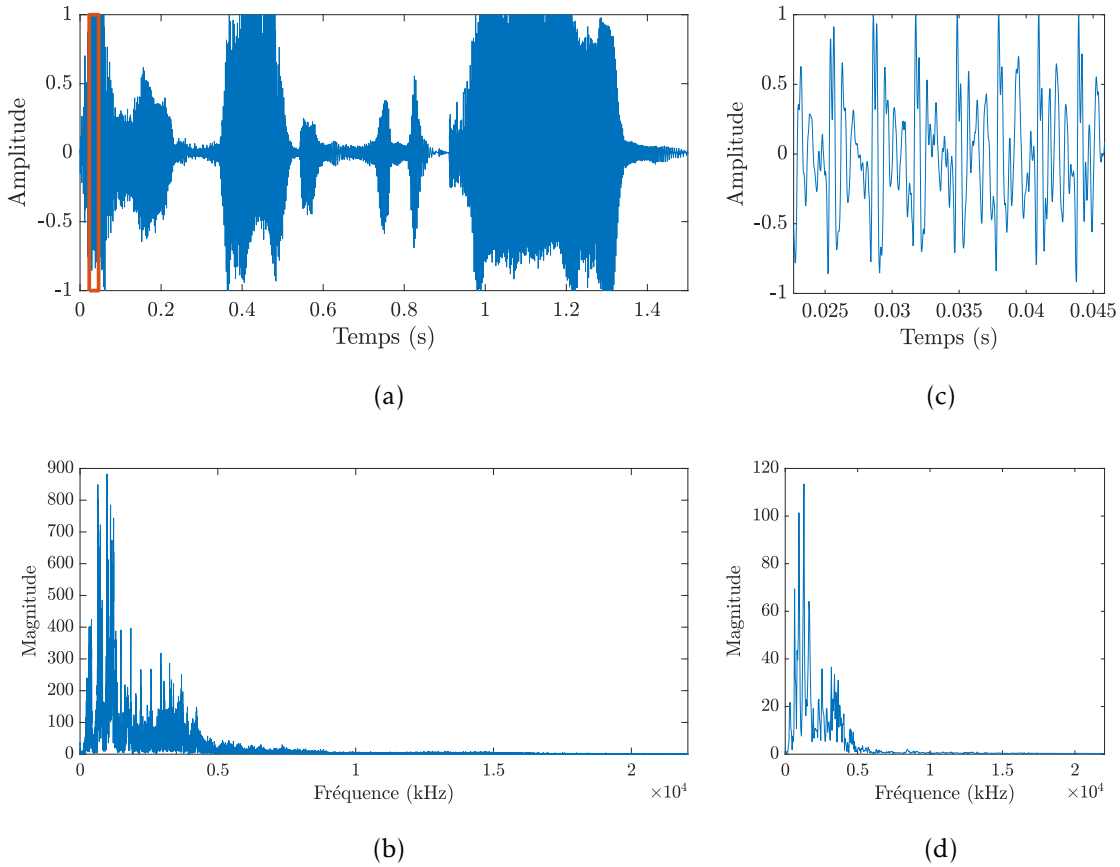


FIGURE 2.2 – Exemple d'un signal de voix en temporel (a) et l'amplitude de sa TFD (b). Une frame est extraite de ce son (c), représenté par le rectangle rouge en (a), et son spectre de puissance associé est calculé (d).

2.1.2 Représentation spectrale des signaux audio numériques

La représentation dans le domaine fréquentiel des signaux est souvent utilisée pour l'analyse des signaux. Cette représentation $\mathbf{x}^{\text{freq}} = \left(x_k^{\text{freq}}\right)_{k=0}^{T-1}$ est calculée en utilisant la *Transformée de* FOURIER *Discrète* (TFD) (voir FIGURE 2.2).

Définition 3 (Transformée de FOURIER Discrète (TFD))**.** *La TFD d'un signal* $\mathbf{x}^{temp}$ *temporel de longueur* T *est définie par :*

$$x_k^{freq} = \sum_{n=0}^{T-1} x_n^{temp} e^{-2\pi i \frac{nk}{T}}. \tag{2.2}$$

On note cette opération $\mathbf{x}^{\text{freq}} = f_{\text{TFD}}(\mathbf{x}^{\text{temp}})$. L'indice de fréquence k définit la discrétisation de l'axe des fréquences : $f_k = k\frac{F_e}{T}$. Le résultat de cette représentation étant un vecteur de nombres complexes, deux informations sont importantes : l'amplitude $\left|x_k^{\text{freq}}\right|$ et la phase $\angle x_k^{\text{freq}}$. Le signal original peut être reconstruit grâce à la *TFD inverse* (TFDI).

Définition 4 (Transformée de Fourier Inverse (TFDI))**.** *La TFDI d'un spectre* $\mathbf{x}^{freq}$ *est définie par :*

$$x_n^{temp} = \frac{1}{T}\sum_{k=0}^{T-1} x_k^{freq} e^{2\pi i \frac{nk}{T}}. \tag{2.3}$$

Si la fréquence d'échantillonnage F_e n'est pas bien choisie, le phénomène de *repliement spectral* ou *aliasing* apparait. Ce phénomène implique qu'une partie du spectre (typiquement les fréquences supérieures à la fréquence d'échantillonnage) va se replier sur les basses fréquences, et donc la reconstruction ne sera pas parfaite. Par exemple, considérons un signal analogique $x^{\text{temp}}(t) = \sin\left(2\pi\frac{F_0}{10}t\right) + \sin\left(2\pi\frac{10F_0}{13}t\right)$ avec $F_0 = 2000$Hz, échantillonné à $F_{e1} = 4000$Hz et $F_{e2} = 2000$Hz. Le signal numérique $x_1(t)$ échantillonné à F_{e1} n'aura pas d'aliasing car la fréquence d'échantillonnage est supérieure à deux fois la plus grande fréquence présente $\left(\frac{10F_0}{13}\right)$. Par contre, pour $x_2^{\text{temp}}(t)$ échantillonné à F_{e2}, la plus grande fréquence va être «translatée» à $F_{e2} - \frac{10F_0}{13}$.

Cette représentation dans le domaine fréquentiel est bonne pour des signaux «statiques» (dont les caractéristiques spectrales ne varient pas beaucoup dans le temps, comme les signaux périodiques), mais dans le cas d'un signal «dynamique» les variations temporelles sont plus difficilement exploitable dans la TFD. C'est pourquoi nous allons présenter des méthodes pour étudier des signaux variant dans le temps, ce qui est utile dans le traitement *temps-réel*.

2.2 Traitement de signal en temps-réel

La plupart des notions relatives au traitement signal audio en temps-réel sont disponibles dans l'oeuvre de Julius O. Smith III [102] [2].

2.2.1 Analyse temps-fréquence d'un signal : Transformée de Fourier à Court Terme

L'analyse temps-fréquence consiste à calculer une représentation dépendant du temps et de la fréquence en même temps. Cette analyse est réalisée en découpant un signal en *frames*, utilisant une fonction de *fenêtrage* et finalement en prenant la TFD de chaque frame fenêtrée.

Une frame est basée sur deux paramètres : Δt le nombre d'échantillons dans la frame et δt le nombre d'échantillons du décalage temporel. Un son peut contenir $N = \lfloor (T - \Delta t)/\delta t \rfloor$ frames. Une frame $\mathbf{x}_\tau^{\text{temp}}$ est définie par :

$$\mathbf{x}_\tau^{\text{temp}} = x^{\text{temp}}_{[t+(\tau-1)\delta t, t+(\tau-1)\delta t+\Delta t]},\ \tau \in \{1,...,N\} \tag{2.4}$$

La frame est fenêtrée en utilisant une fenêtre $\mathbf{w} = (\mathrm{w}_n)_{n=0}^{\Delta t-1}$ de taille Δt :

$$(\mathbf{w}\cdot\mathbf{x}_\tau^{\text{temp}})_n = \mathrm{w}_n x_{\tau n}^{\text{temp}},\ n = 0,...,\Delta t - 1, \tag{2.5}$$

avec $\cdot$ le produit élément par élément. Finalement la TFD de la frame fenêtrée est calculée :

$$\mathbf{x}_\tau^{\text{freq}} = f_{\text{TFD}}\left(\mathbf{w}\cdot\mathbf{x}_\tau^{\text{temp}}\right). \tag{2.6}$$

2. Ces ressources sont également disponibles en ligne : https://ccrma.stanford.edu/~jos/sasp/

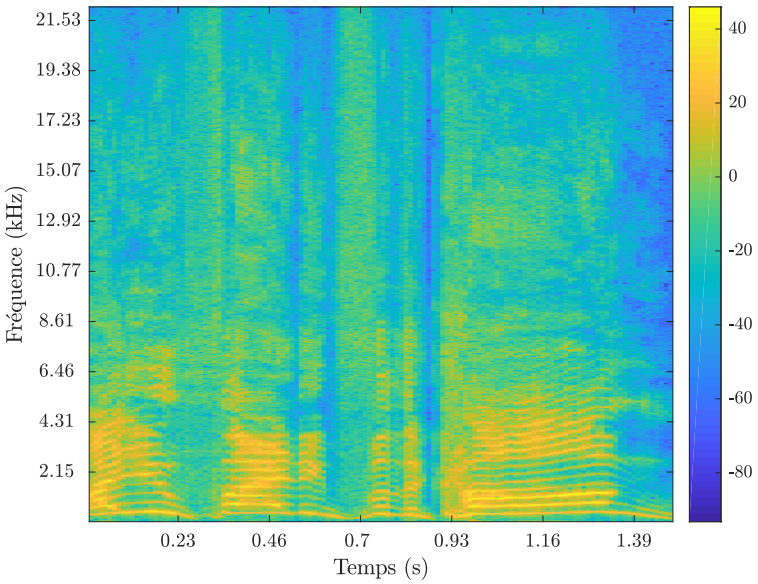


FIGURE 2.3 – Exemple de spectrogramme d'un signal de parole. Les couleurs correspondent à l'intensité (en dB) dans chaque bin temps-fréquence (τ, k) (bleu = faible, jaune = fort). Le signal est échantillonné à $F_e = 44,1$kHz, la fenêtre utilisée est une fenêtre de Hanning de taille $\Delta t = 1024$ échantillons et le décalage est de $\delta t = 512$ échantillons.

Le résultat $\left(\mathbf{x}_\tau^{\text{freq}}\right)_{\tau=1}^{N}$ est appelé la *Transformée de* FOURIER *à Court Terme* (TFCT) de $\mathbf{x}^{\text{temp}}$. La (log-) amplitude de la TFCT est appelée le (log-)*spectrogramme* du signal et peut être visualisée sous la forme d'une image (voir FIGURE 2.3).

Le fenêtrage est particulièrement utile dans le cas de *l'analyse-synthèse*. Dans le cas de l'analyse de signaux sonores il permet de réduire les lobes secondaires comparés à une fenêtre rectangulaire. En effet le spectre d'une porte temporelle est un sinus cardinal fréquentiel comportant énormément de lobes secondaires, contrairement à une fenêtre de HANNING par exemple (voir plus bas). Dans le cas de la synthèse, le recouvrement entre différentes fenêtres permet de lisser les variations temporelles d'une frame à l'autre. Le choix de la fenêtre $\mathbf{w}$ dépend du traitement à effectuer [102]. En effet l'opération de fenêtrage correspond à une convolution dans le domaine fréquentiel et donc le spectre observé n'est pas le même que celui de la frame avant fenêtrage. Les fenêtres suivantes sont illustrées en FIGURE 2.4 :

- Fenêtre *rectangulaire* : $\mathrm{w}_n = 1,\ n = 0,...,\Delta t - 1$. Le principal problème de cette fenêtre est la présence de grands lobes secondaires dans le spectre.
- Fenêtre de *Hanning* : $\mathrm{w}_n = 0,5\left(1 - \cos\left(\frac{2\pi n}{\Delta t - 1}\right)\right),\ n = 0,...,\Delta t - 1$. Cette fenêtre possède des lobes secondaires plus faibles que la porte et est donc plus appropriée pour le traitement audio.
- Fenêtre de *Hamming* : $\mathrm{w}_n = 0,54 - 0,46\cos\left(\frac{2\pi n}{\Delta t - 1}\right),\ n = 0,...,\Delta t - 1$. Cette fenêtre a encore moins de lobes secondaires que la fenêtre de Hanning.

Habituellement le décalage δt est choisi comme une fraction de Δt. Par exemple ce décalage peut être de $\delta t = \Delta t/2$: on a ainsi un overlap de 50%. De la même manière, $\delta t = \Delta t/4$ est un overlap de 75% et $\delta t = (3/4)\Delta t$ est un overlap de 25% (voir FIGURE 2.5).

Dans le cadre d'un traitement temps-fréquence comme la TFCT, le découpage en frame

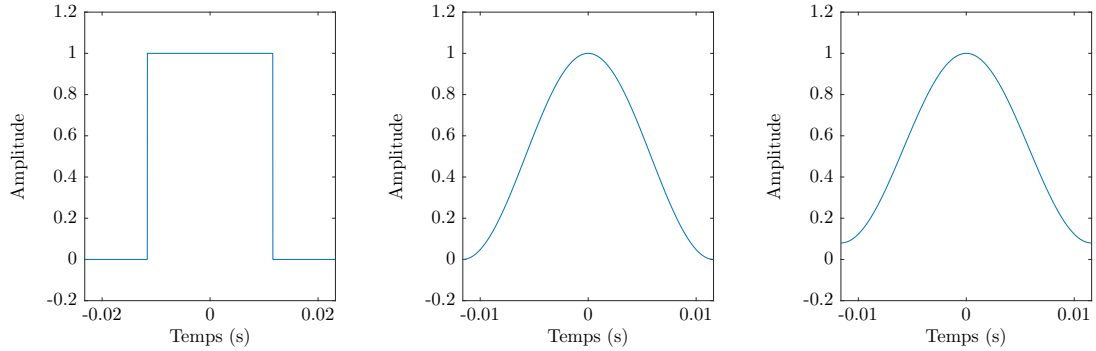


FIGURE 2.4 – Exemples de fenêtres : rectangulaire (a), Hanning (b) et Hamming (c). Elles sont toutes de taille $\Delta t = 1024$ échantillons pour un échantillonnage à $F_e = 44,1$kHz.

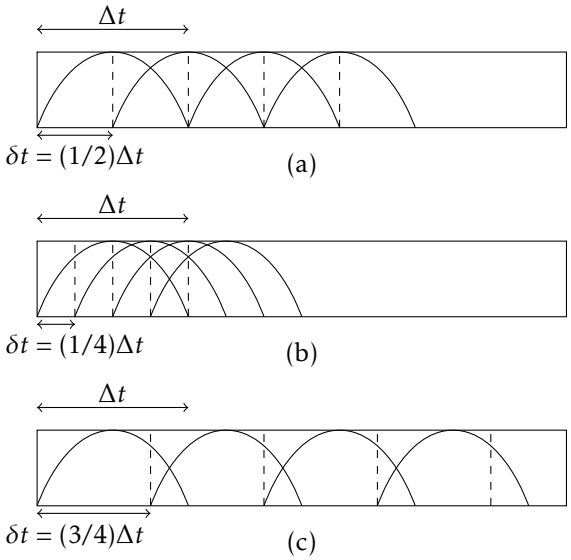


FIGURE 2.5 – Illustration de l'overlap entre différentes frames : 50% (a), 75% (b) et 25% (c). Chaque fenêtre a une taille $\Delta t = 1024$ et le décalage varie de respectivement $\delta t = (1/2)\Delta t$, $(1/4)\Delta t$, $(3/4)\Delta t$.

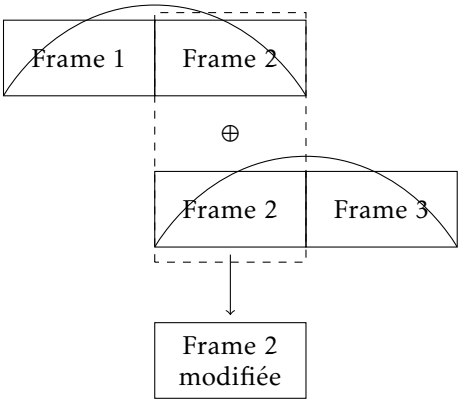


FIGURE 2.6 – Principe de l'overlap-add. On considère un buffer (ou tampon en français) de deux fenêtres que l'on traite. Ensuite on décale d'une fenêtre pour avoir un nouveau buffer que l'on traite. La frame numéro 2 est donc traitée deux fois : il suffit de sommer ces deux frames pour obtenir la frame 2 finale.

du signal permet l'analyse sur des portions quasi-stationnaires de ce signal. Dans un cadre temps-réel, il peut y avoir des soucis de continuité entre les différentes trames au moment d'une étape de reconstruction. Pour pallier ce problème, on utilise la technique de *l'overlap-add* (voir FIGURE 2.6). Celle-ci consiste à prendre des blocs de deux frames qui sont fenêtrés et sur lesquels on applique le traitement. On décale ensuite d'une frame et ce nouveau bloc de deux frames est fenêtré puis traité. On remarque qu'une frame a été traitée deux fois (en tant que nouvelle frame et ancienne frame), et la reconstruction consiste à sommer ces deux frames : ainsi la continuité est assurée. Pour obtenir une reconstruction parfaite avec un recouvrement de 50% dans l'overlap-add, la fenêtre d'analyse doit vérifier la propriété suivante :

$$\mathrm{w}_n^2 + \mathrm{w}_{n+\Delta t/2}^2 = 1,\ n = 0,...\Delta t/2 + 1. \tag{2.7}$$

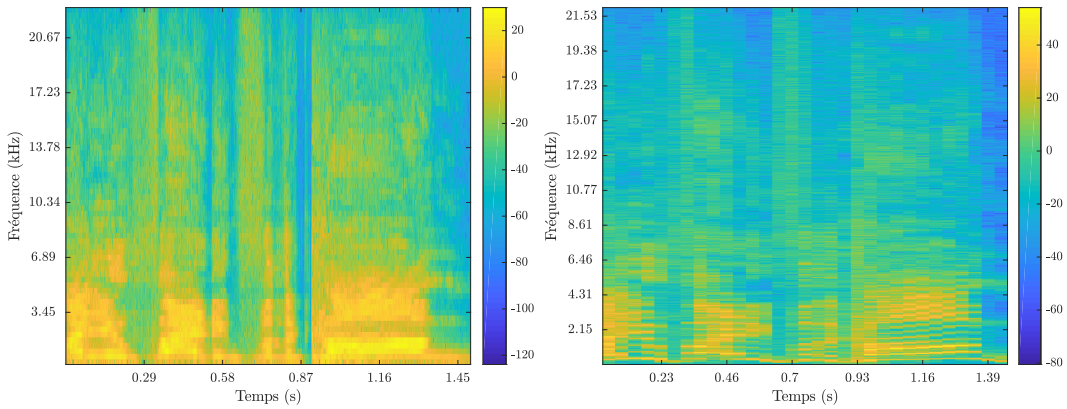
En pratique la racine de la fenêtre de HANNING sera donc considérée lorsqu'il y aura des étapes d'overlap-add (notamment en séparation).

D'autres types d'analyse temps-fréquence peuvent être considérées, comme la *Transformée en ondelettes* par exemple [66]. Celle-ci est basée sur une décomposition multirésolution du signal suivant une base d'ondelettes.

L'analyse temps-fréquence est sujette au *principe d'incertitude*, aussi connu sous le nom du théorème de GABOR :

Théorème 2 (Principe d'incertitude [117])**.** *Il est impossible d'avoir à la fois une bonne localisation dans le domaine temporel et dans le domaine fréquentiel :* $\Delta\tau\Delta f \geq \frac{1}{4\pi}$ *avec* $\Delta\tau$ *et* Δf *les écart-types des distributions du domaine temporel et fréquentiel du signal* $\mathbf{x}^{temp}$*, respectivement.*

Cela implique qu'il faut faire un choix suivant la situation : soit le traitement est bien localisé en fréquence soit il est bien localisé en temps. Le choix aura un impact sur la forme du spectrogramme (voir FIGURE 2.7).



(a) (b)

FIGURE 2.7 – Exemples de deux spectrogrammes avec différentes tailles de fenêtres : bonne localisation en temps (a) avec $\Delta t = 128$ échantillons, et bonne localisation en fréquence (b) avec $\Delta t = 4096$ échantillons. Les couleurs correspondent à l'intensité (en dB) dans chaque bin temps-fréquence (bleu = faible, jaune = fort).

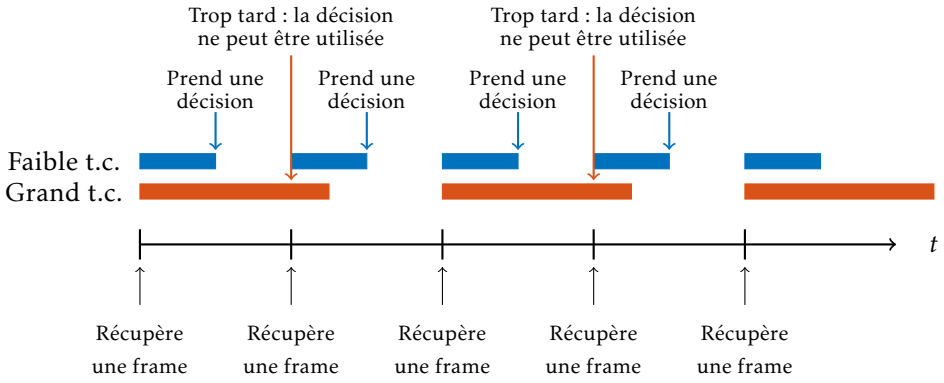


FIGURE 2.8 – Illustration de la latence d'un système temps-réel de classification audio.

2.2.2 Caractéristiques du traitement de signal temps-réel

Le concept de temps-réel fait référence à deux critères [34] : la *vitesse* et la *latence*. La *vitesse* est le temps mis pour prendre une décision : elle est reliée au nombre de frames utilisées par le système. Par exemple, un système rapide n'utilisera que peu de frames pour prendre sa décision (1 ou 2 par exemple). Ensuite, la *latence* est reliée au temps de calcul (voir FIGURE 2.8). Par exemple, si une frame dure 50ms, alors la décision doit être calculée en moins de 50ms, sinon elle ne pourra pas être utilisée dans le processus.

Une illustration de la latence est disponible sur la FIGURE 2.8. Un processus temps-réel récupère des frames à des instants spécifiques (flèches noires). Si le temps de calcul (t.c.) est faible (rectangle bleu plein), la décision pourra être utilisée par le système car elle sera accessible avant la prochaine frame. Si le temps de calcul est grand (rectangle rouge plein), la décision ne pourra pas être utilisée par le système. Le choix de la vitesse et de la latence dépend des contraintes imposées par l'utilisateur.

2.3 Théorie de l'apprentissage statistique

2.3.1 Présentation générale

L'apprentissage statistique est une famille de méthodes et d'algorithmes servant à apprendre des modèles et extraire de l'information à partir de données brutes [14, 39]. Le but est de trouver une fonction $\mathcal{R}$ entre les entrées $\mathbf{x} \in \mathbb{R}^d$, appelées les *attributs*, et la sortie $\mathbf{y}$ telle que $\mathbf{y} = \mathcal{R}(\mathbf{x})$. Cette sortie peut être un vecteur réel (on parle alors de régression) ou le label d'une classe (on parle alors de classification). L'apprentissage statistique est basé sur deux étapes principales : *l'extraction d'attributs* et un *algorithme d'apprentissage*.

L'extraction d'attributs consiste à extraire des informations pertinentes des données brutes qui serviront à l'algorithme d'apprentissage, à l'aide d'une fonction $f(\cdot)$. Par exemple, les descripteurs audio sont utilisés en classification audio [84], les sacs de mots pour le traitement du langage naturel, *etc*. C'est une étape importante car elle conditionne les performances de l'algorithme d'apprentissage : celui-ci pourrait ne pas apprendre les bonnes informations et donc ne pourrait pas généraliser son apprentissage. L'étape d'extraction d'attributs est très dépendante du domaine (audio, texte, image, *etc.*). Dans le cas de la grande dimension (beaucoup de données en entrée, à la fois en taille d'échantillon et en nombre d'attributs), une étape de *réduction de dimension*

peut être envisagée, soit de manière supervisée (avec une analyse discriminante de Fisher [39] par exemple, si on dispose d'échantillons labellisés) ou non supervisée (avec une Analyse en Composantes Principales (PCA) [14] par exemple).

Un algorithme d'apprentissage utilise un ensemble de données dit *d'apprentissage* $\left(\mathbf{x}_{(i)}, \mathbf{y}_{(i)}\right)_{i=1}^{n}$ (avec n le nombre d'exemples, ou échantillons statistiques, disponibles) pour estimer la fonction $\mathcal{R}$ en optimisant une *fonction de perte* $\mathcal{L}$ qui est une perte moyenne entre les vraies valeurs de la sortie $\mathbf{y}_{(i)}$ et les valeurs prédites $\widehat{\mathbf{y}}_{(i)} = \mathcal{R}(\mathbf{x}_{(i)})$ sur l'ensemble d'apprentissage. La fonction de perte $\mathcal{L}$ est importante et dépend de l'objectif de l'apprentissage. Dans le cas d'une régression, une fonction de perte classique est la *moyenne des écarts au carré* (MSE) .

Définition 5 (Moyenne des écarts au carré (MSE))**.** *La moyenne des écarts au carré est l'espérance du carré de la différence entre la vraie valeur et la valeur prédite :*

$$\mathcal{L}_{\mathrm{MSE}}(\mathbf{y}, \mathcal{R}(\mathbf{x})) = \mathbb{E}\left[\mathbf{y} - \mathcal{R}(\mathbf{x}) \middle| \mathbf{x}\right]^2. \tag{2.8}$$

La formule empirique correspondante est :

$$\widehat{\mathcal{L}}_{\mathrm{MSE}}\left(\left(\mathbf{y}_{(i)}, \mathcal{R}\left(\mathbf{x}_{(i)}\right)\right)_{i=1}^{n}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{y}_{(i)} - \mathcal{R}\left(\mathbf{x}_{(i)}\right)\right)^2. \tag{2.9}$$

Dans le cas de la classification binaire (où on cherche à prédire une classe ou son contraire) la fonction de perte usuelle est *la fonction de coût 0-1* (0-1).

Définition 6 (Fonction de coût 0-1 (0-1))**.** *La fonction de coût 0-1 est définie par l'espérance du nombre de fois que la prédiction est fausse :*

$$\mathcal{L}_{0-1}(\mathbf{y}, \mathcal{R}(\mathbf{x})) = \mathbb{E}\left[\mathbf{1}(\mathbf{y} \neq \mathcal{R}(\mathbf{x})) \middle| \mathbf{x}\right], \tag{2.10}$$

avec $\mathbf{1}(\mathbf{y} \neq \mathcal{R}(\mathbf{x})) = 1$ *si* $\mathbf{y} \neq \mathcal{R}(\mathbf{x})$ *et 0 sinon. La formule empirique correspondante est :*

$$\widehat{\mathcal{L}}_{0-1}\left(\left(\mathbf{y}_{(i)}, \mathcal{R}\left(\mathbf{x}_{(i)}\right)\right)_{i=1}^{n}\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\left(\mathbf{y}_{(i)} \neq \mathcal{R}\left(\mathbf{x}_{(i)}\right)\right). \tag{2.11}$$

La classification peut être binaire (deux classes) ou multi-classes (plus de deux classes), mais également uni-label (un seul label caractérise l'échantillon) ou multi-labels (plusieurs labels simultanés caractérisent l'échantillon). La classification multi-classes multi-labels étant complexe, plusieurs méthodes permettent de simplifier le multi-labels en un problème uni-label [4]. La première est la *Binary Relevance* (BR) et elle consiste en un apprentissage de chaque classe séparément : l'avantage de cette méthode est qu'elle est simple mais elle n'apprend pas les corrélations posssibles entre les classes. La seconde méthode est le *Label Powerset* (LP) et elle consiste à considérer une instance multi-labels comme une nouvelle classe (par exemple dans une classification à 3 classes, si une combinaison des classes 1 et 2 est présente, elle sera considérée comme une nouvelle classe, labellisée 4 par exemple) : cette méthode est plus complexe que la précédente et introduit un aspect combinatoire important dans le cas où le nombre de classes est grand. Il existe également des méthodes multi-labels plus avancées telles que RAKEL [113] (combinaison de classifieurs LP) ou les chaînes de classifieurs [90] (chaîne de classifieurs BR) et leurs versions d'ensemble.

Les prochaines sous-parties présentent des méthodes classiques d'apprentissage statistique qui sont utilisées pour de la classification et/ou de la séparation de sources sonores (cette dernière pouvant être considérée comme une tâche de régression).

2.3.2 Modèles de mélanges

Définition

La première méthode considérée est une méthode de classification appelée *modèle de mélanges* qui peut être utilisée à la fois dans un cas d'apprentissage *non supervisé* ou *supervisé*. Le début de cette partie explique la théorie sur les modèles de mélanges dans un cas non supervisé puis son utilisation dans un cas supervisé est donnée en fin de partie.

Les modèles de mélanges sont des modèles génératifs de données où la distribution des classes et la distribution des données sachant les classes sont explicitées. On note $\mathbf{X} \in \mathbb{R}^d$ le vecteur aléatoire qui représente les attributs et $\mathbf{Z} = (Z_1, ..., Z_G) \in \mathbb{R}^G$ le vecteur aléatoire qui représente le label de la *composante du mélange* – c'est un vecteur à entrées binaires tel que $Z_g = 1$ si $\mathbf{X}$ appartient à la classe g, 0 sinon – et G le nombre de composantes. C'est une variable dite *latente* dans le cas général car souvent celle-ci n'est pas disponible en pratique. Le modèle génératif suppose que les échantillons sont *indépendants et identiquement distribués* (i.i.d.) suivant le processus suivant :

— Tirage d'un label de classe : $\mathbf{Z} \sim \text{Mult}_G(\cdot; 1, \boldsymbol{\pi})$,

— Tirage d'un échantillon suivant la classe : $\mathbf{X}|Z_g = 1 \sim p\left(\cdot; \boldsymbol{\theta}_g\right)$,

avec $\text{Mult}_G(\cdot; n, \boldsymbol{\pi})$ est la loi multinomiale sur G cases à n tirages avec probabilités $\boldsymbol{\pi} = \left(\pi_g\right)_{g=1}^G$ telle que $\pi_g > 0$, $\sum_{g=1}^G \pi_g = 1$, d'expression :

$$\text{Mult}_G(\mathbf{z}; n, \boldsymbol{\pi}) = \frac{n!}{\prod_{g=1}^G z_g!} \prod_{g=1}^G \pi_g^{z_g}, \tag{2.12}$$

et $p\left(\mathbf{x}; \boldsymbol{\theta}_g\right)$ la distribution des données dans la classe g de vecteur de paramètres $\boldsymbol{\theta}_g$. La densité de $\mathbf{X}$, dite *densité mélange*, peut alors s'écrire :

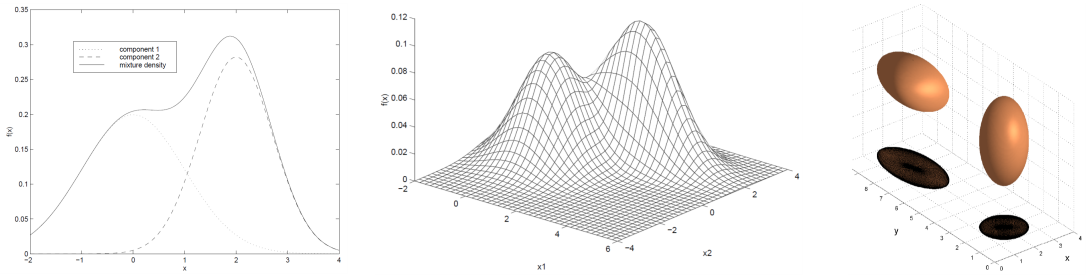
$$p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{g=1}^G \pi_g p\left(\mathbf{x}; \boldsymbol{\theta}_g\right), \tag{2.13}$$

avec $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_G\}$ les paramètres de la distribution. Un exemple de composante possible pour le modèle de mélanges est la *distribution gaussienne* :

$$\begin{aligned} p\left(\mathbf{x}; \boldsymbol{\theta}_g\right) &= \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_g, \Sigma_g\right) && (2.14) \\ &= (2\pi)^{-d/2} \left(\det\left(\Sigma_g\right)\right)^{-1/2} \exp\left(-\frac{1}{2}\left(\mathbf{x} - \boldsymbol{\mu}_g\right)^\top \Sigma_g^{-1} \left(\mathbf{x} - \boldsymbol{\mu}_g\right)\right), && (2.15) \end{aligned}$$

avec $\boldsymbol{\theta}_g = \left\{\boldsymbol{\mu}_g, \Sigma_g\right\}$ les paramètres de la distribution normale : $\boldsymbol{\mu}_g \in \mathbb{R}^d$ le vecteur moyenne (et $\boldsymbol{\mu}_g^\top$ sa transposée) et $\Sigma_g \in \mathbb{R}^{d \times d}$ la matrice de covariance, et $\mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_g, \Sigma_g\right)$ la densité gaussienne prise en $\mathbf{x}$. On parle dans ce cas de *modèle de mélanges gaussiens* (GMM). Des exemples de composantes gaussiennes sont disponibles à la Figure 2.9.

Une autre composante importante est la *distribution de* Dirichlet pour des vecteurs $\mathbf{x}$ tels que $\sum_{j=1}^d x_j = 1$:



(a) (b) (c)

FIGURE 2.9 – Exemple de composantes gaussiennes en 1D (a), 2D (b) et 3D (c) (extrait de [12]).

$$p\left(\mathbf{x};\boldsymbol{\theta}_g\right) = \mathrm{Dir}\left(\mathbf{x};\boldsymbol{\alpha}_g\right) = \frac{\Gamma\left(\sum_{j=1}^{d}\alpha_{gj}\right)}{\prod_{j=1}^{d}\Gamma\left(\alpha_{gj}\right)}\prod_{j=1}^{d}x_j^{\alpha_{gj}-1}, \tag{2.16}$$

avec $\boldsymbol{\theta}_g = \boldsymbol{\alpha}_g \in \mathbb{R}^d$ le vecteur de paramètre tel que $\alpha_{gj} > 0$, et $\mathrm{Dir}\left(\mathbf{x};\boldsymbol{\alpha}_g\right)$ la densité prise en $\mathbf{x}$. On parle dans ce cas de *modèle de mélanges de* DIRICHLET (DMM).

Les échantillons de l'ensemble d'apprentissage sont supposés i.i.d, ce qui permet de définir la vraisemblance des paramètres de la distribution.

Définition 7 (Vraisemblance des paramètres)**.** *Sous hypothèse que les* $\mathbf{x}_{(i)}$ *sont i.i.d., on définit la vraisemblance des paramètres par :*

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} p\left(\mathbf{x}_{(i)};\boldsymbol{\theta}\right). \tag{2.17}$$

On note également la log-vraisemblance :

$$\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}). \tag{2.18}$$

Estimation

Les paramètres $\boldsymbol{\theta}$ de la distribution peuvent être estimés en optimisant la vraisemblance : les paramètres les plus «probables» sachant les données observées sont alors estimés. C'est *l'estimation par maximum de vraisemblance*.

Définition 8 (Estimateur de maximum de vraisemblance)**.** *L'estimateur du maximum de vraisemblance des données observées est la valeur de* $\boldsymbol{\theta}$ *qui maximise cette log-vraisemblance :*

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\mathrm{argmax}}\,\ell(\boldsymbol{\theta}). \tag{2.19}$$

Dans le cas des modèles de mélanges, l'optimisation de cette vraisemblance est souvent réalisée avec *l'algorithme EM* (Espérance - Maximisation), présenté historiquement par DEMPSTER *et al.* [24] dans un cadre non-supervisé. C'est un algorithme itératif qui maximise la log-vraisemblance des données complètes en deux étapes.

Définition 9 (Vraisemblance des données complètes)**.** *La vraisemblance des données complètes est définie comme la densité jointe des attributs et des variables latentes sur chaque échantillon par la formule suivante (sous hypothèse que les couples* $(\mathbf{x}_{(i)}, \mathbf{z}_{(i)})$ *sont i.i.d.) :*

$$L_c(\boldsymbol{\theta}) = \prod_{i=1}^{n} p\left(\mathbf{x}_{(i)}, \mathbf{z}_{(i)}; \boldsymbol{\theta}\right). \tag{2.20}$$

On note également la log-vraisemblance des données complètes :

$$\ell_c(\boldsymbol{\theta}) = \log L_c(\boldsymbol{\theta}). \tag{2.21}$$

À l'itération (t) l'algorithme se décompose en deux étapes :

(Étape E) Calculer $Q\left((\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}\right) = \mathbb{E}\left[\ell_c(\boldsymbol{\theta}) \middle| \mathbf{x}_{(1)}, ..., \mathbf{x}_{(n)}; \boldsymbol{\theta}^{(t)}\right]$,
(Étape M) Maximiser $Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}\right) : \boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}\right)$.

L'algorithme EM augmente la valeur de la vraisemblance à chaque itération mais souvent il ne converge que vers un maximum local de la vraisemblance. Dans le cas d'un GMM, l'algorithme EM prend la forme suivante. L'étape E consiste à calculer les espérances des variables latentes :

$$\overline{z}_{ig}^{(t)} = \mathbb{E}^{(t)}\left[z_{ig} \middle| \mathbf{x}_{(i)}; \boldsymbol{\theta}^{(t)}\right] = \frac{\pi_g \mathcal{N}\left(\mathbf{x}_{(i)}; \boldsymbol{\mu}_g, \Sigma_g\right)}{\sum_{g'=1}^{G} \pi_{g'} \mathcal{N}\left(\mathbf{x}_{(i)}; \boldsymbol{\mu}_{g'}, \Sigma_{g'}\right)}. \tag{2.22}$$

L'étape M consiste à estimer les nouveaux paramètres :

$$\widehat{\pi}_g^{(t+1)} = \frac{\sum_{i=1}^{n} \overline{z}_{ig}^{(t)}}{n}, \tag{2.23}$$

$$\widehat{\boldsymbol{\mu}}_g^{(t+1)} = \frac{\sum_{i=1}^{n} \overline{z}_{ig}^{(t)} \mathbf{x}_{(i)}}{\sum_{i=1}^{n} \overline{z}_{ig}^{(t)}}, \tag{2.24}$$

$$\widehat{\Sigma}_g^{(t+1)} = \frac{\sum_{i=1}^{n} \overline{z}_{ig}^{(t)} \left(\mathbf{x}_{(i)} - \widehat{\boldsymbol{\mu}}_g^{(t+1)}\right)\left(\mathbf{x}_{(i)} - \widehat{\boldsymbol{\mu}}_g^{(t+1)}\right)^{\top}}{\sum_{i=1}^{n} \overline{z}_{ig}^{(t)}}. \tag{2.25}$$

Les mélanges gaussiens peuvent également être estimés sur des *données par intervalle* [67]. Dans ce cas, l'espace d'échantillonnage $\mathcal{X}$ de $\mathbf{X} = \left(\mathbf{X}_{(1)}, ..., \mathbf{X}_{(n)}\right)$ est divisé en r sous-ensembles mutuellement exclusifs $\mathcal{X}_j$. De plus, on ne connait que le nombre n_j d'observations de $\mathbf{X}_{(i)}$ tombant dans chaque espace $\mathcal{X}_j$. On note $n = \sum_{j=1}^{r} n_j$, et on suppose que $\mathbf{y} = (n_1, ..., n_r)$ suit une loi multinomiale de n tirages sur r catégories, de probabilités $P_j(\boldsymbol{\theta})/P(\boldsymbol{\theta})$, avec :

$$\begin{aligned} P_j(\boldsymbol{\theta}) &= \int_{\mathcal{X}_j} p(\mathbf{x}|\boldsymbol{\theta}) \mathrm{d}\mathbf{x}, \\ P(\boldsymbol{\theta}) &= \sum_{j=1}^{r} P_j(\boldsymbol{\theta}). \end{aligned} \tag{2.26}$$

La log-vraisemblance s'écrit dans ce cas :

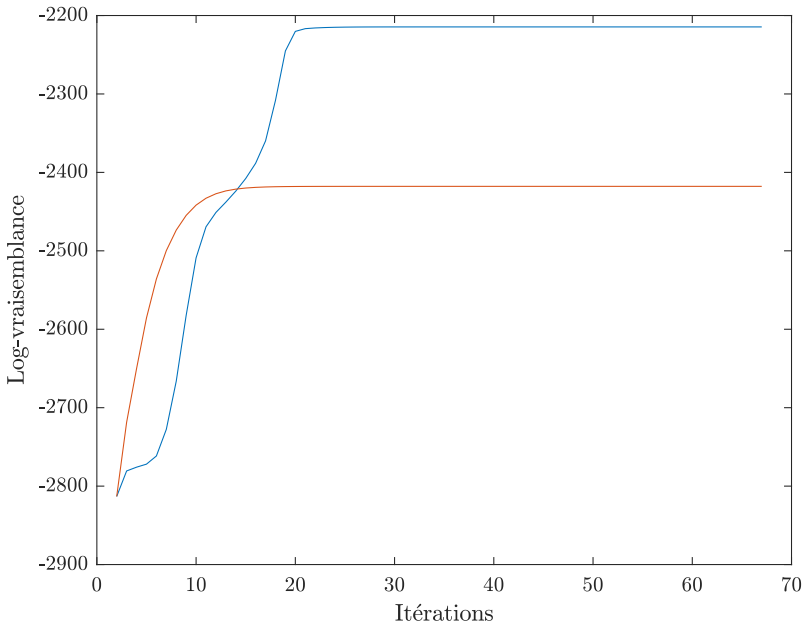


FIGURE 2.10 – Différences de convergence dans l'estimation d'un mélange gaussien dans le cas classique (ligne rouge) et dans le cas par intervalle (ligne bleue).

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{y}) = \left(\frac{n!}{\prod_{j=1}^{r} n_j!}\right) \prod_{j=1}^{r} \left[\frac{P_j(\boldsymbol{\theta})}{P(\boldsymbol{\theta})}\right]^{n_j}. \tag{2.27}$$

L'estimation des paramètres se fait avec un algorithme EM comme décrit en Annexe B. Du fait du nombre plus important de données manquantes dans le schéma par intervalle, la convergence de l'algorithme est plus longue que dans le cas standard. Par exemple, pour un mélange gaussien à trois composantes en 1D, la FIGURE 2.10 montre l'évolution de la log-vraisemblance dans le cas standard et dans le cas par intervalle.

De plus, dans le cas des DMM, l'estimation des paramètres est assez complexe à cause de la forme de la vraisemblance. Plusieurs méthodes existent comme *l'inférence variationnelle* dont MA *et al.* ont proposé une procédure d'estimation dans [64]. Une estimation par algorithme EM et optimisation de type NEWTON-RAPHSON a également été proposée par MA *et al.* dans [63]. Des éléments de détails concernant ces procédures sont disponibles dans l'Annexe B.

Sélection de modèles

L'estimation d'un modèle de mélange $\mathbf{m} = \{p(\mathbf{x};\boldsymbol{\theta}); \boldsymbol{\theta} \in \Theta\}$ nécessite de définir le nombre de composantes G qui composent le mélange. Ce nombre peut être estimé «à l'oeil» ou grâce à un *a priori* sur le problème (nombre de classes connu à l'avance). Des techniques plus rigoureuses existent pour permettre son estimation, basées sur une optimisation d'un critère particulier. Le but de ces critères est de pénaliser la log-vraisemblance par un terme qui mesure la complexité du modèle.

Tableau 2.1 – Illustration de la sélection de modèle par AIC et BIC dans l'estimation d'un mélange gaussien à 3 composantes.

Critère	Nombre de composantes 1	2	3	4	5	Optimum
AIC	5643	5311	4340	**4339**	4344	4
BIC	5653	5336	**4380**	4393	4413	3

On peut formaliser ces critères sous la forme :

$$\mathrm{crit}(\widehat{\mathbf{m}}) = -2\ell\left(\widehat{\boldsymbol{\theta}}\right) + \mathrm{pen}(\widehat{\mathbf{m}}). \tag{2.28}$$

On retiendra le modèle dont le critère est *minimum*. Un des critères les plus connus est le *critère d'information d'Akaike* (AIC) [3], où la pénalité représente le nombre de paramètres continus ν dans le modèle.

Définition 10 (Critère AIC)**.** *Le critère d'information d'Akaike (AIC) est défini par :*

$$\mathrm{AIC}(\widehat{\mathbf{m}}) = -2\ell\left(\widehat{\boldsymbol{\theta}}\right) + 2\nu. \tag{2.29}$$

Néanmoins les conditions de régularités nécessaires à l'emploi de ce critère ne sont pas toujours vérifiées dans le cas des modèles de mélanges. En pratique ce critère surestime le nombre de composantes dans le mélange [104]. Un autre critère très connu est le *critère d'information bayésien* (BIC), développé par Schwarz [97]. C'est une approximation de la vraisemblance intégrée par la méthode de Laplace.

Définition 11 (Critère BIC)**.** *Le critère d'information Bayésien (BIC) est défini par :*

$$\mathrm{BIC}(\widehat{\mathbf{m}}) = -2\ell\left(\widehat{\boldsymbol{\theta}}\right) + \nu \log n. \tag{2.30}$$

Contrairement à l'AIC, le BIC ne surestime pas le nombre de composantes dans le mélange asymptotiquement. Une illustration de l'influence du critère sur le choix du nombre de composantes est disponible dans le Tableau 2.1. Le même mélange gaussien à 3 composantes que précédemment est estimé pour un nombre de composantes allant de 1 à 5 et les critères AIC et BIC sont calculés pour chaque cas. On remarque que le critère AIC choisit 4 composantes (surestime) alors que le critère BIC en choisit 3 (le vrai nombre).

Les modèles de mélanges sont un outil puissant de modélisation car toutes les hypothèses de génération de données sont explicitées et permettent une meilleure interprétabilité des données.

Cas supervisé

Dans un cas de classification supervisé, il est possible d'utiliser les GMM, de la manière suivante. Au sein de chaque classe z, un GMM contenant $G^{(z)}$ composantes est appris, de densité $p\left(\mathbf{x}\middle|z;\widehat{\boldsymbol{\theta}}^{(z)}\right)$. Une fois les modèles de mélanges appris, il est possible de s'en servir pour construire une règle de décision $\mathcal{R}$. Dans le cas de la classification, il suffit de considérer la règle du *maximum a posteriori* (MAP), c'est-à-dire :

$$\widehat{z} = \widehat{\mathcal{R}}(\mathbf{x}) = \operatorname*{argmax}_z p\left(z\middle|\mathbf{x};\widehat{\boldsymbol{\theta}}^{(z)}\right), \tag{2.31}$$

où $p\left(z\middle|\mathbf{x};\widehat{\boldsymbol{\theta}}^{(z)}\right)$ est calculé avec la règle de Bayes, on peut écrire :

$$p\left(z\middle|\mathbf{x};\widehat{\boldsymbol{\theta}}^{(z)}\right) \propto p\left(\mathbf{x}\middle|z;\widehat{\boldsymbol{\theta}}^{(z)}\right)p\left(z;\widehat{\boldsymbol{\theta}}^{(z)}\right). \tag{2.32}$$

Packages informatiques

L'implémentation numérique de cette méthode est disponible dans des librairies de programmation standards. Sous MATLAB [3], la classe `gmdistribution` permet de créer un modèle de mélange gaussien dont les paramètres sont les vecteurs moyennes `mu`, les matrices de covariances `sigma` et les proportions `p`. L'apprentissage est réalisé grâce à `fitgmdist` en lui fournissant les données d'entrées `X` et le nombre de composantes du mélange. Une fois le modèle créé et appris, l'évaluation de la densité en un point se fait avec la méthode `gmdistribution.pdf`.

Il existe une alternative *open source* dans le package `scikit-learn` écrit en Python [4]. Ce package contient un module `mixture` où la classe `GaussianMixture` peut être utilisée pour créer un mélange, l'apprendre et évaluer la densité.

Chacune de ces librairies dispose également d'options supplémentaires pour spécifier la forme de la matrice de covariance (`full` pour une matrice complète, `diag` pour ne considérer qu'une matrice diagonale, *etc.*), la méthode d'initialisation des paramètres (complètement aléatoire ou avec un *k-means*) et bien d'autres options. Il est également possible de faire de la sélection de modèles car la plupart de ces librairies calculent de manière automatique les critères usuels de sélection de modèles.

2.3.3 Machines à Vecteurs de support

Les machines à vecteurs de support (SVM) sont une classe de méthodes qui peut servir à la régression ou à la classification [7]. Elles sont basées sur le principe de *classifieur linéaire à vaste marge*, et sont principalement utilisées dans le cas de classification binaire (les classes étant notées $y_1 = +1$ et $y_2 = -1$). Un classifieur linéaire est défini par un hyperplan de paramètres $\mathbf{w} \in \mathbb{R}^d$ et $b \in \mathbb{R}$ de la manière suivante :

$$\mathbf{w}^\top \mathbf{x} + b \begin{cases} \geq 0 & \text{si } y = +1, \\ < 0 & \text{sinon.} \end{cases} \tag{2.33}$$

avec $\mathbf{w}^\top$ le vecteur transposé de $\mathbf{w}$ [5]. Une particularité des SVM est d'augmenter la marge par rapport à la frontière de décision $\mathbf{w}^\top \mathbf{x} + b = 0$ du classifieur linéaire pour le rendre robuste au fait que la frontière entre les classes n'est pas un hyperplan. On décide par exemple que le point $\mathbf{x}_+$ de la classe +1 le plus proche de la frontière de décision satisfait $\mathbf{w}^\top \mathbf{x}_+ + b = 1$, et le point $\mathbf{x}_-$ de la classe -1 satisfait $\mathbf{w}^\top \mathbf{x}_- + b = -1$.

La distance de l'origine à un point $\mathbf{x}$ dans la direction $\mathbf{w}$ est définie par $\mathbf{w}^\top \mathbf{x}/\sqrt{\mathbf{w}^\top \mathbf{w}}$. Ainsi la marge entre les deux hyperplans séparant les deux classes est $2/\sqrt{\mathbf{w}^\top \mathbf{w}}$. Le problème à optimiser revient donc à minimiser la longueur $\mathbf{w}^\top \mathbf{w}$ de manière à maximiser la marge :

$$\min_{\mathbf{w}} \tfrac{1}{2}\mathbf{w}^\top \mathbf{w} \tag{2.34}$$

$$\text{s.c. } \mathbf{y}_{(i)}\left(\mathbf{w}^\top \mathbf{x}_{(i)} + 1\right) \geq 1,\ i = 1, ..., n. \tag{2.35}$$

3. https ://fr.mathworks.com/help/stats/gmdistribution.html
4. http ://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html
5. À ne pas confondre avec la fenêtre d'analyse $\mathbf{w}$ en traitement de signal.

où s.c. signifie sous contrainte. La contrainte de marge peut être relâchée en ajoutant une variable «ressort» $\xi_{(i)}$ de manière à prendre en compte des données qui ne sont pas vraiment linéairement séparables, de sorte que le problème d'optimisation se réécrit alors :

$$\min_{\mathbf{w}} \tfrac{1}{2}\mathbf{w}^\top\mathbf{w} + \tfrac{C}{2}\|\xi\| \tag{2.36}$$

$$\text{s.c. } \mathbf{y}_{(i)}\left(\mathbf{w}^\top\mathbf{x}_{(i)} + 1\right) \geq 1 - \xi_{(i)},\ i = 1,...,n. \tag{2.37}$$

C'est un problème de programmation quadratique, qui se résout très bien en utilisant la formulation des multiplicateurs de Lagrange. En écrivant la formulation duale, on voit apparaître le produit scalaire entre chaque $\mathbf{x}_{(i)}$ et $\mathbf{x}_{(j)}$ que l'on note $K\left(\mathbf{x}_{(i)},\mathbf{x}_{(j)}\right) = \mathbf{x}_{(i)}^\top\mathbf{x}_{(j)}$. L'astuce du noyau (ou *kernel trick* en anglais) peut alors être utilisée : on remplace le produit scalaire classique en «envoyant» les données d'origine dans un espace de plus grande dimension $\mathcal{H}$ via la fonction $\phi : \mathbb{R}^d \to \mathcal{H}$ qui vérifie $K\left(\mathbf{x}_{(i)},\mathbf{x}_{(j)}\right) = \left\langle \phi\left(\mathbf{x}_{(i)}\right), \phi\left(\mathbf{x}_{(j)}\right)\right\rangle_{\mathcal{H}}$ avec $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ le produit scalaire dans $\mathcal{H}$. Cette astuce utilise la *théorie des noyaux semi-définis positifs* et les espaces de Hilbert à noyaux reproduisant noté $\mathcal{H}$. C'est la grande force des SVM car cela permet de classifier des données qui ne sont pas linéairement séparables en les envoyant dans un espace de plus grande dimension où elles le seront, sans connaître la fonction ϕ.

La SVM étant une méthode de classification binaire, il est nécessaire de repenser l'apprentissage dans le cas multi-classes [120]. Par exemple, des stratégies différentes d'entrainement ont été considérées :

— OVO (One-Versus-One) : cette stratégie consiste à entrainer des SVM binaires pour toutes les paires de classes possibles, ce qui demande $K \times (K-1)/2$ classifieurs. Chaque SVM estime la classe la plus probable et donne donc un vote, et la règle de décision globale consiste en un vote majoritaire.

— OVA (One-Versus-All) : cette stratégie consiste à entrainer des SVM binaires où une classe est considérée comme +1 et les autres -1, ce qui demande $K-1$ classifieur. Le label de classe sera estimé grâce au classifieur dont la valeur de sortie est la plus grande.

Plutôt que d'estimer des SVM binaires séparément avec les stratégies précédentes, des auteurs ont considéré des procédés d'optimisation pour apprendre des SVM multiclasses directement comme Weston et Watkins [122] ou Crammer et Singer [23]. Ces méthodes proposent d'estimer les SVM binaires ensemble en modifiant la fonction objectif et les contraintes.

De même que pour les GMM, des implémentations numériques des SVM sont disponibles dans les librairies standards. Sous MATLAB, il est possible d'apprendre un SVM binaire [6] via `fitcsvm` ou multi-classes [7] via `fitcecoc`. Il existe une alternative *open source* dans le package `scikit-learn` écrit en Python [8]. Ce package contient un module `SVM` où la classe `SVC` peut être utilisée pour créer un SVM multi-classes. Il est notamment possible de sélectionner la stratégie d'apprentissage multi-classes (OVO ou OVA).

6. https ://fr.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html
7. https ://fr.mathworks.com/help/stats/fitcecoc.html
8. http ://scikit-learn.org/stable/modules/svm.html

2.3.4 Décomposition en matrices non-négatives

Définition

La décomposition en matrices non-négatives est une famille de méthodes qui cherche à décomposer une matrice non-négative en un produit de deux matrices de rang inférieur à la matrice d'origine [65, Chapitre 1] [33]. La décomposition en matrices non-négatives est à la base une technique non supervisée mais il est possible de l'utiliser dans un cas supervisé (comme expliqué plus loin).

Définition 12 (Factorisation en matrices non-négatives (NMF))**.** *Soit* $\mathbf{X} \in \mathbb{R}^{n\times d}$ *une matrice non-négative (c'est-à-dire dont les entrées sont positives ou nulles), on définit la* factorisation en matrices non-négatives *(NMF) de* $\mathbf{X}$ *par :*

$$\mathbf{X} = \mathbf{WH}, \tag{2.38}$$

avec $\mathbf{W} \in \mathbb{R}^{n\times k}$ *et* $\mathbf{H} \in \mathbb{R}^{k\times d}$ *et* $k < n, d$ *le rang des matrices de la factorisation.*

On note $\mathbf{X}_{ij}$ l'élément de $\mathbf{X}$ correspondant à la i^e ligne et j^e colonne. Un exemple de décomposition de spectrogramme de piano avec la NMF est disponible à la Figure 2.11.

Remarque. Une extension de la NMF aux tenseurs est également disponible, appelé NTF (Nonnegative Tensor Factorization). Un tenseur est un tableau multidimensionnel, dont le vecteur et la matrice sont deux cas particuliers. Des exemples de tenseurs sont par exemple les spectrogrammes de modulation [8] ou le spectrogramme multicanal.

Estimation

L'estimation de ces deux matrices se fait en optimisant une *divergence* entre la matrice d'origine et le produit des deux matrices. Plusieurs divergences existent, dont la divergence de Kullback-Leibler et la divergence de Itakura-Saito, qui font partie de la famille des divergences de Bregman.

Définition 13 (Divergence de Kullback-Leibler)**.** *La divergence de* Kullback-Leibler *entre* $\mathbf{X}$ *et* $\mathbf{WH}$ *est définie par :*

$$d_{\mathrm{KL}}(\mathbf{X}, \mathbf{WH}) = \sum_{i,j}\left(\mathbf{X}_{ij}\log\frac{\mathbf{X}_{ij}}{(\mathbf{WH})_{ij}} - \mathbf{X}_{ij} + (\mathbf{WH})_{ij}\right). \tag{2.39}$$

Définition 14 (Divergence de Itakura-Saito)**.** *La divergence d'*Itakura-Saito *est définie par :*

$$d_{\mathrm{IS}}(\mathbf{X}, \mathbf{WH}) = \sum_{i,j}\left(\frac{\mathbf{X}_{ij}}{(\mathbf{WH})_{ij}} - \log\frac{\mathbf{X}_{ij}}{(\mathbf{WH})_{ij}} - 1\right). \tag{2.40}$$

L'optimisation de ces deux divergences amène à différentes règles de mises à jour des matrices. La plus commune est la règle de *mise à jour multiplicative* (MU). Dans le cas de la divergence d'Itakura-Saito, les règles suivantes sont à appliquer de manière itérative jusqu'à convergence :

$$\begin{aligned}\mathbf{H} &\leftarrow \mathbf{H}\cdot\frac{\mathbf{W}^\top\left((\mathbf{WH})^{\cdot -2}\cdot\mathbf{X}\right)}{\mathbf{W}^\top(\mathbf{WH})^{\cdot -1}},\\ \mathbf{W} &\leftarrow \mathbf{W}\cdot\frac{\left((\mathbf{WH})^{\cdot 2}\cdot\mathbf{X}\right)\mathbf{H}^\top}{(\mathbf{WH})^{\cdot -1}\mathbf{H}^\top}.\end{aligned} \tag{2.41}$$

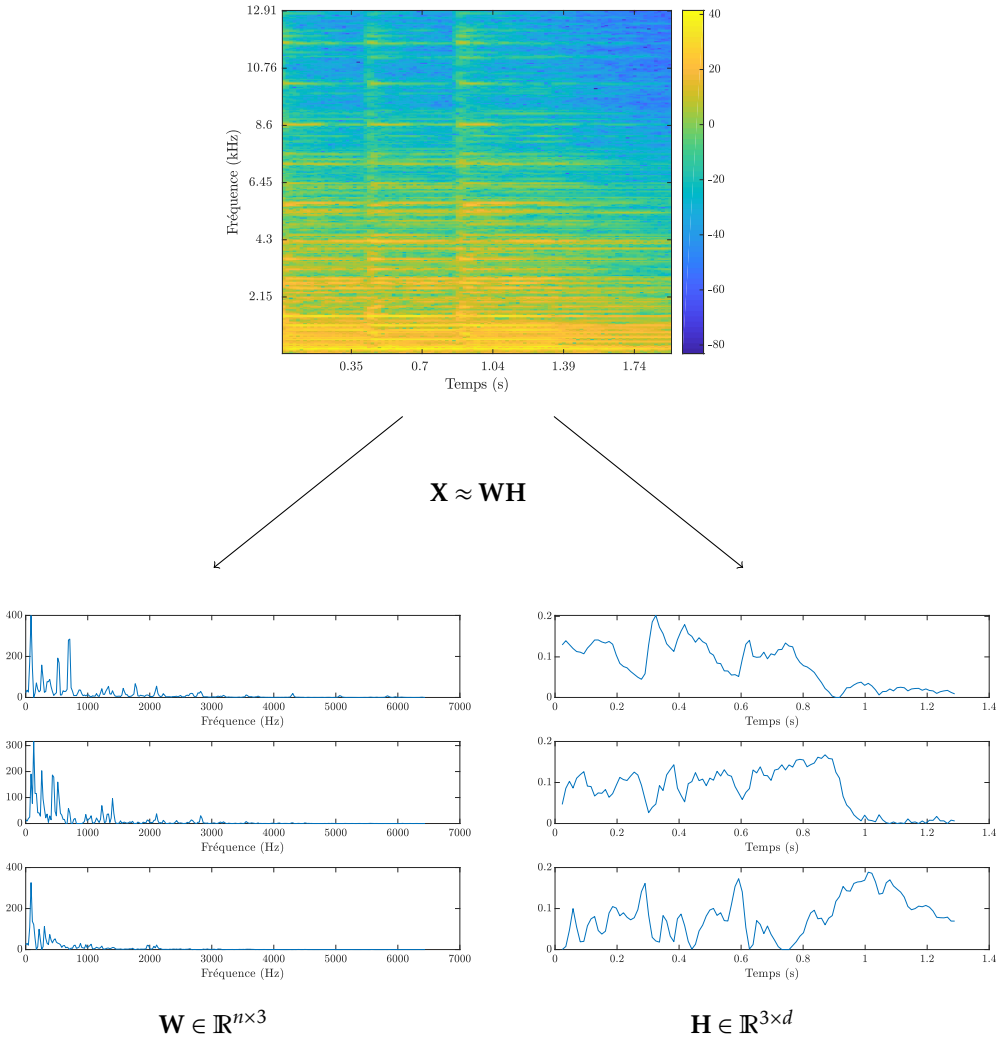


FIGURE 2.11 – Exemple de décomposition d'un spectrogramme de piano **X** avec une NMF à 3 composantes. A gauche les bases spectrales et à droite les activations temporelles.

avec $\cdot$, $\frac{\cdot}{\cdot}$, et $\mathbf{A}^{\cdot b}$ la multiplication, division et puissance élément par élément, respectivement. En pratique la divergence d'Itakura-Saito est la plus utilisée pour la séparation de sources comme montré par Févotte dans [33].

Une méthode probabiliste proche de la NMF a été développée par Smaragdis et s'appelle *l'Analyse probabiliste en composantes latentes* (PLCA, Probabilistic Latent Component Analysis en anglais) [65, Chapitre 3]. Ce modèle suppose que la matrice $\mathbf{X}$ possède des entrées non-négatives et également entières. L'idée est alors de chercher la distribution bivariée qui a généré ces données sous la forme :

$$P(f,t) = \sum_g P(z=g)P(f|z=g)P(t|z=g) \tag{2.42}$$

avec z les composantes latentes du modèle. Ce modèle a surtout été utilisé pour de la séparation de sources sonores, c'est pourquoi il fait apparaître les variables fréquence (f) et temps (t) du spectrogramme qu'il cherche à factoriser. Un algorithme EM permet d'estimer les différentes composantes temporelles et fréquentielles de cette distribution. À l'étape E il s'agit de calculer :

$$P(z|f,t) = \frac{P(z)P(f|z)P(t|z)}{\sum_{z'} P(z')P(f|z')P(t|z')}. \tag{2.43}$$

L'étape M consiste à mettre à jour les distributions :

$$P(z) = \frac{\sum_f \sum_t \mathbf{X}_{ft} P(z|f,t)}{\sum_{z'} \sum_f \sum_t \mathbf{X}_{ft} P(z'|f,t)}, \tag{2.44}$$

$$P(f|z) = \frac{\sum_t \mathbf{X}_{ft} P(z|f,t)}{\sum_{f'} \sum_t \mathbf{X}_{f't} P(z|f',t)}, \tag{2.45}$$

$$P(t|z) = \frac{\sum_f \mathbf{X}_{ft} P(z|f,t)}{\sum_f \sum_{t'} \mathbf{X}_{ft'} P(z|f,t')}. \tag{2.46}$$

Dans la formulation ci-dessus, la PLCA et la NMF font une séparation non supervisée des sources sonores. Une version supervisée de ces méthodes requiert d'apprendre d'abord les composantes fréquentielles sur des sources d'intérêt ($\mathbf{W}$ ou $P(f|z=k)$), puis au moment de séparer il suffit d'estimer les activations ($\mathbf{H}$ ou $P(t|z)$) en fixant les composantes fréquentielles.

Packages informatiques

Plusieurs librairies de programmation existent pour réaliser un apprentissage d'une NMF. Sous MATLAB, il est possible d'apprendre une NMF [9] via `nnmf` en lui spécifiant le nombre de composantes. Il existe une alternative open source dans le package `scikit-learn` écrit en Python [10]. Ce package contient un module `decomposition` où la classe `NMF` peut être utilisée pour créer une NMF. La PLCA quant à elle n'était pas disponible, j'ai donc du la coder en partant de zéro.

9. https ://fr.mathworks.com/help/stats/nnmf.html
10. http ://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html

2.3.5 Apprentissage profond (Deep Learning)

Définition

L'apprentissage profond (ou *Deep Learning* en anglais) est devenu incontournable en apprentissage statistique ces dernières années [56], grâce à sa capacité de généralisation dans des domaines comme la reconnaissance d'image ou de la parole. Il est basé sur l'utilisation de *réseaux de neurones artificiels* qui sont basiquement une manière d'organiser des calculs complexes en réseaux de calculs par graphe. Le principe général est l'approximation de n'importe quelle opération grâce aux non-linéarités présentes dans le réseau.

Le point de départ des réseaux de neurones est le *perceptron*, c'est-à-dire une couche composée d'un neurone unique qui fonctionne de la manière suivante. On note $\mathbf{w} \in \mathbb{R}^d$ et $b \in \mathbb{R}$ les paramètres du perceptron, et $\sigma(\cdot)$ une fonction dite *d'activation* (le plus souvent non linéaire). Un perceptron réalise l'opération suivante :

$$y = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right). \tag{2.47}$$

Dans la formulation ci-dessus, le perceptron se rapproche d'une régression logistique. L'intérêt des réseaux de neurones est de concaténer plusieurs neurones au sein d'une couche, puis de concaténer ces couches de neurones pour apprendre des formes complexes dans les données. Les réseaux de neurones sont ainsi souvent utilisés pour de *l'apprentissage de représentation*. Les différentes fonctions d'activations possibles sont les suivantes :

— *Sigmoide*. C'est la fonction d'activation historique des réseaux de neurones car elle est censée imiter le comportement d'un neurone biologique. Elle est définie par :

$$\sigma(x) = \frac{1}{1+\exp(-x)}. \tag{2.48}$$

Elle a la propriété de renvoyer une valeur entre 0 et 1 et sa dérivée s'exprime avec $\sigma' = \sigma(1-\sigma)$.

— *Tangente hyperbolique*. Elle est définie par :

$$\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}. \tag{2.49}$$

Elle a la propriété de renvoyer une valeur entre -1 et 1 et sa dérivée s'exprime avec $\tanh' = 1 - \tanh^2$.

— *ReLU* (Rectified Linear Unit). Elle est définie par :

$$\mathrm{ReLU}(x) = \max(x, 0). \tag{2.50}$$

La particularité de cette fonction est sa rapidité d'exécution et sa dérivée très simple, qui en ont fait la fonction la plus utilisée dans les réseaux complexes (les réseaux convolutionnels notamment) car les temps d'apprentissage sont fortement réduits.

En utilisant plusieurs perceptrons au sein d'une couche neuronale on construit la base du *réseau de neurones à propagation avant* (FNN), illustré en Figure 2.12, qui se formalise de la manière suivante :

$$\mathbf{h}^{(\ell)} = \sigma\left(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \tag{2.51}$$

où la fonction d'activation est appliquée élément par élément, ℓ le numéro de la couche cachée,

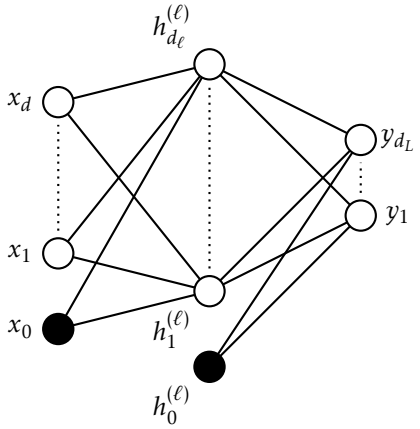


FIGURE 2.12 – Schéma d'un réseau de neurones (inspiré de [14]). Le réseau est constitué d'une couche d'entrée contenant les attributs $\mathbf{x} = (x_0, x_1, ..., x_d)$ (avec x_0 le biais), une couche cachée $\mathbf{h}^{(\ell)}$ ($\ell = 1$) et une couche de sortie $\mathbf{y}$. Le graphe se lit de gauche à droite : un noeud de la couche ℓ est connecté aux noeuds de la couche précédente et ce noeud représente l'opération de combinaison linéaire avec non linéarité (similaire à Eq. (2.47)).

$\mathbf{h}^{(\ell)} = \left[h_1^{(\ell)}, ..., h_{d_\ell}^{(\ell)}\right] \in \mathbb{R}^{d_\ell}$ les neurones de la couche ℓ, $\mathbf{W}^\ell = \left[\mathbf{w}_1^\ell, ..., \mathbf{w}_{d_\ell}^\ell\right] \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ les poids qui lient la couche ℓ et $\ell - 1$ et $\mathbf{b}^{(\ell)}$ le biais. Le cas particulier de $\ell = 0$ correspond à la couche d'entrée ($\mathbf{h}^{(0)} = \mathbf{x}$) et celui de $\ell = L$ à celui de la sortie ($\mathbf{h}^L = \mathbf{y}$). Si le réseau ne possède qu'une couche cachée ($L = 1$) on parle de FNN. Dans le cas de plus de deux couches cachées on parle de *réseau de neurones profond* (DNN).

Deux autres structures de réseaux ont émergé récemment, les *réseaux de neurones convolutionnels* (CNN) et les *réseaux de neurones récurrents* (RNN). Les CNN sont particulièrement utilisés pour le traitement des images. En effet les calculs effectués par les CNN prennent en compte la dimension spatiale via les convolutions contrairement aux DNN classiques. L'idée générale du CNN est d'apprendre des attributs intéressants issus des spectrogrammes via des couches convolutionnelles. Entre chaque couche, une étape de *pooling* est réalisée pour diminuer la dimension des attributs appris par le réseau. Pour éviter le surapprentissage, des couches *dropout* sont également insérées entre chaque couche convolutionnelle. Le dropout consiste à désactiver certains neurones de manière aléatoire dans le réseau : il a été montré que cette procédure permet de limiter le surapprentissage [105]. Les dernières couches sont des couches denses et finissent par une couche softmax qui permet de calculer des scores interprétés comme des probabilités de présence de chaque classe. Les RNN sont plutôt utilisés dans le cas des séries temporelles car ils permettent de garder une certaine «mémoire» des évènements passés. Deux principales cellules récurrentes sont utilisées : les *unités récurrentes fermées* (GRU) et les *mémoires longues à court-terme* (LSTM, Long Short-Term Memory).

Estimation

L'apprentissage des réseaux de neurones se fait en optimisant une fonction de coût, le plus souvent par *descente de gradient stochastique* (SGD) avec *rétropropagation des erreurs* : c'est l'un des algorithmes les plus efficaces pour optimiser de telles fonctions mathématiques.

La SGD est une version randomisée de la descente de gradient [17]. Cette dernière consiste à

optimiser une fonction de manière itérative en mettant à jour les paramètres grâce au gradient de la fonction. Si on note $\theta = \left(\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\right)_{\ell=1}^{L}$ tous les paramètres du réseau, $\mathcal{L}(\theta)$ la fonction de coût associée au réseau et $\nabla\mathcal{L}(\theta)$ le gradient de cette fonction, la mise à jour des paramètres à l'itération $(t+1)$ se fait à partir des paramètres de l'itération (t) avec la formule suivante :

$$\theta^{(t+1)} = \theta^{(t)} - \eta\nabla\mathcal{L}\left(\theta^{(t)}\right), \tag{2.52}$$

où η est appelé *taux d'apprentissage* : il contrôle la vitesse à laquelle la descente de gradient converge. L'inconvénient de la descente de gradient classique est que dans le cas de grandes masses de données le calcul du gradient sur tout l'ensemble d'apprentissage peut être prohibitif. La SGD est une version randomisée de la descente de gradient où seulement un échantillon de l'ensemble d'apprentissage tiré au hasard est utilisé pour mettre à jour les paramètres. Elle s'adapte bien dans le cas de grandes bases de données mais la convergence n'est pas aussi rapide qu'avec la descente de gradient. Une version intermédiaire consiste à prendre des groupes (*batch*) d'échantillons pour mimer la SGD tout en gardant des propriétés de la descente de gradient.

La rétropropagation des erreurs [92] est une méthode permettant d'évaluer le gradient de manière efficace dans le cas des réseaux de neurones en utilisant la règle de dérivée en chaîne. Elle utilise la structure particulière des réseaux de neurones pour calculer les dérivées partielles du gradient en rétropropageant les termes de la sortie vers l'entrée.

Packages informatiques

Plusieurs packages de programmation existent pour réaliser un apprentissage d'un DNN. Sous MATLAB, les possibilités sont limitées mais il existe quelques fonctions pour apprendre des réseaux de neurones pour la classification d'image notamment [11]. Il est plus simple de faire du Deep Learning avec des packages Python, notamment Tensorflow [12], Keras [13] ou PyTorch [14]. Chacun des packages Python a ses propres avantages et inconvénients :

- Keras est une surcouche à Tensorflow qui permet de créer des réseaux de neurones très rapidement et simplement en spécifiant uniquement l'entrée, les couches intermédiaires et la sorties ainsi que la méthode d'optimisation,
- Tensorflow est une librairie créée et utilisée par Google rendue open source récemment, et permet de faire des développements assez complexes, ce qui en fait une librairie plus difficile à appréhender que Keras,
- PyTorch est en quelque sorte un compromis entre Tensorflow et Keras car il permet d'appréhender rapidement les réseaux de neurones tout en laissant le choix à l'utilisateur de coder des fonctions plus complexes.

2.3.6 Apprentissage par transfert

L'apprentissage par transfert [82] est une partie de l'apprentissage statistique qui a connu ses débuts basé sur le constat suivant. En apprentissage statistique «classique», les données d'apprentissage et de test sont supposées être tirées de la même distribution. Si une différence entre les deux apparait, il est souvent nécessaire de refaire un apprentissage pour obtenir un nouveau modèle. L'apprentissage par transfert entend résoudre ce problème de différences entre données apprentissage / test. Ce type d'apprentissage apparait dans la littérature sous différents

11. https ://fr.mathworks.com/help/deeplearning/ug/deep-learning-in-matlab.html
12. https ://www.tensorflow.org/
13. https ://keras.io/
14. https ://pytorch.org/

noms, comme l'apprentissage longue durée, le transfert de connaissance, le transfert inductif et bien d'autres.

Le formalisme de l'apprentissage par transfert se base sur un *domaine* $\mathcal{D} = \{\mathcal{X}, \mathbb{P}(\mathbf{x})\}$ comprenant l'espace des entrées $\mathcal{X}$ et la distribution de probabilité de $\mathbf{x}$ ainsi qu'une *tâche* $\mathcal{T} = \{\mathcal{Z}, \mathcal{R}(\mathbf{x})\}$ comprenant l'espace des sorties $\mathcal{Z}$ (ici les labels) et la fonction de prédiction $\mathcal{R}(\cdot)$. L'idée est de pouvoir passer d'un domaine et une tâche source, notés $\mathcal{D}_S$ et $\mathcal{T}_S$, à un domaine et une tâche cible, notés $\mathcal{D}_T$ et $\mathcal{T}_T$, grâce à une fonction de prédiction cible $\mathcal{R}_T$ apprise sur le domaine source. Il faut donc répondre à plusieurs questions :

— «Quoi» transférer, c'est-à-dire quelles parties du domaine source peuvent être utilisées pour le domaine cible,
— «Quand» transférer, c'est-à-dire dans quelles situations et surtout quand il ne faut pas transférer : par exemple lorsque les deux domaines sont trop éloignés l'un de l'autre,
— «Comment» transférer, c'est-à-dire en utilisant quelle méthode.

L'apprentissage par transfert peut être découpé en trois grandes méthodes qui sont l'apprentissage par transfert *inductif*, *transductif* et *non-supervisé*. Le transfert inductif étudie le cas où la tâche cible est différente de la tâche source (peu importe les domaines source et cible). Le transfert transductif étudie le cas où le domaine cible est différent du domaine source mais où la tâche source est la même que la tâche cible. Le transfert non supervisé étudie le cas où la tâche cible est différente mais a un lien avec la tâche source.

L'apprentissage par transfert peut être utilisé dans le cas de la classification audio [116] par exemple.

2.4 Conclusion

Dans ce chapitre les différentes notions utiles à la compréhension du reste du manuscrit ont été développées. Le traitement numérique du signal est au coeur de la problématique soulevée par cette thèse, et la solution considérée utilise l'apprentissage statistique.

Les principales méthodes présentées dans ce chapitre serviront de base de comparaison avec nos modélisations probabilistes : nous verrons notamment que les techniques d'apprentissage profond ne sont pas aussi performantes que notre méthode dans le cas de la classification en temps-réel. À l'inverse, des méthodes assez standards comme les GMM obtiennent des résultats assez convenables.

La science est la classification
systématique de l'expérience.

George Henry Lewes

Chapitre 3

Classification de sources audio en temps-réel

Ce chapitre présente la méthode de classification de sources sonores en temps-réel développée durant la thèse. Un état de l'art des descripteurs audio usuels ainsi que des méthodes de classification audio est disponible en partie 3.2 et partie 3.3 respectivement. La méthode développée, appelé RARE pour *Real-Time Recognition Engine*, est déclinée suivant les trois problèmes qu'elle cherche à résoudre : classification monophonique en partie 3.4, classification polyphonique en partie 3.5 et passage au temps-réel en partie 3.6. Cette méthode couvre à la fois la classification monophonique (une source présente) et polyphonique (plusieurs sources) grâce à un modèle génératif des spectres de puissances des sources et une estimation non paramétrique par noyaux. Les expériences montrent en partie 3.7 que la méthode possède plusieurs avantages comparée aux méthodes de l'état de l'art. La partie 3.8 conclut le chapitre et résume les avancées majeures du travail réalisé ici.

3.1 Introduction

La classification de sources sonores est un sujet de recherche stimulant depuis plusieurs décennies : ses prémices se trouvent en outre dans la reconnaissance de la parole dans les années 80 [88] et on la rencontre de nos jours sous le nom de détection et classification d'évènements sonores (SED).

Le but de cette dernière est donc de détecter et classifier des sources sonores présentes dans des flux *monophoniques* (une source présente à la fois) et *polyphoniques* (plusieurs sources). Beaucoup de méthodologies et d'algorithmes ont été créés pour résoudre ce problème, et peuvent être regroupés dans trois sujets principaux. Le sujet le plus connu est la reconnaissance vocale automatique (ASR) dont le but est d'identifier la parole (en particulier les phonèmes) dans un enregistrement audio [87, 91, 112]. L'ASR est notamment utilisée dans les assistants personnels type Google Home, Siri, Cortana, *etc.* ou encore les services vocaux de certains standards téléphoniques. Le sujet suivant est la recherche d'informations musicales [16, 46] qui vise à analyser des musiques et à en extraire des informations pertinentes telles que le genre musical [53] (rock, classique, *etc*.) ou les différents instruments [45]. Le dernier sujet est la reconnaissance de sons environnementaux qui vise à reconnaître les sons tels que les avions, les toux, les trains, les coups de feu, *etc.* [85, 94].

Cette classification peut être exécutée *hors ligne*, en utilisant l'ensemble du signal, ou *en ligne*, c'est-à-dire que les données audio arrivent au système de classification à la volée sous forme de *frames temporelles*. Le traitement en ligne est aussi appelé *temps-réel*, qui a été défini dans la partie 2.2.

La classification de sources sonores utilise des techniques d'apprentissage statistique, et donc également des attributs pour entrainer ces méthodes. Les attributs utilisés en classification audio sont souvent appelés descripteurs audio et sont présentés dans la partie 3.2. Les techniques de l'état de l'art sur la classification audio sont présentées dans la partie 3.3.

3.2 Descripteurs audio

La classification audio utilise généralement des attributs spécifiques appelés *descripteurs audio* [84]. Ces descripteurs sont extraits des différentes représentations des signaux :

- *Temporelle* : le son brut dépendant du temps est utilisé pour calculer les descripteurs temporels. Ces derniers comportent notamment l'énergie, les coefficients d'autocorrélation et le taux de passage par zéro.
- *Spectrale* : une représentation fréquentielle ou locale de type Transformée de Fourier à court terme (TFCT) est utilisée. Ces descripteurs comportent notamment les moments spectraux (centroid, spread, skewness, kurtosis).
- *Harmonique* : le son est modélisé par un modèle harmonique (somme de sinusoides) plus bruit. Les descripteurs harmoniques comportent notamment la fréquence fondamentale, l'inharmonicité et l'énergie harmonique.
- *Cepstrale* : ils sont basés sur le cepstre du son, c'est-à-dire la TF de la log-amplitude de la TF du son.
- *Perceptuelle* : le son est décomposé suivant une autre échelle de fréquence, censée représenter plus fidèlement l'audition humaine, comme l'échelle Mel, Bark ou ERB (Equivalent Rectangular Bandwidth).

La liste complète des descripteurs est disponible en Annexe A. Ils peuvent être calculés en utilisant tout le signal, ou frame par frame. Les descripteurs à la frame sont le plus souvent utilisés, même dans le cas d'une classification hors ligne. En effet les descripteurs à la frame sont plus robustes aux changements brusques du signal et permettent de capturer des informations plus fines que les descripteurs globaux. Il est également possible d'étudier les variations temporelles de descripteurs spectraux, harmoniques, cepstraux et perceptifs en considérant les dérivées suivant le temps de ces descripteurs [45].

Les descripteurs les plus utilisés dans le cas de la classification sont l'énergie du signal, les MFCC (Mel Frequency Cepstral Coefficients, c'est-à-dire les coefficients du cepstre en échelle Mel), et la fréquence fondamentale (dans le cas de transcription de notes de musique). D'autres descripteurs sont également possibles mais moins courants. Certains auteurs ont utilisé des réseaux de neurones comme des autoencodeurs pour apprendre les attributs [123, 58], ou bien un système joint d'extraction d'attributs et de classification [18]. Une étape de sélection d'attributs peut être considérée pour réduire la dimension : soit par une discriminante de Fisher [45], un blanchiment de données par ACP [94] ou sur une classification hiérarchique basée sur différentes échelles temporelles [93].

Ces descripteurs peuvent être pertinents la plupart du temps mais ils ne sont pas forcément bien adaptés pour tous les types de sons. De plus ils induisent intrinsèquement une perte d'information lors de leur calcul, ce qui peut être préjudiciable lors d'un procédé d'analyse-synthèse audio (transformation puis restitution) et de classification de sources. La méthode de

classification développée dans la thèse propose une transformation minimale sur les données de manière à rester le plus bas niveau possible.

Certaines méthodes de classification audio comme les réseaux de neurones ont besoin de beaucoup de données pour que l'apprentissage soit efficace. Or les bases de données audio ne comportent pas forcément beaucoup d'exemples, notamment dans le cas polyphonique. À titre de comparaison, une base de données audio standard peut contenir quelques centaines de millieurs d'exemples contre plusieurs millions pour une base de données d'images. Dans ce cas, une étape *d'augmentation de données* est nécessaire. Celle-ci consiste à créer de nouveaux échantillons d'apprentissage à partir des échantillons déjà existants. Takahashi *et al.* [110] proposent de mixer aléatoirement des sons d'une même classe en les décalant temporellement et en altérant leur contenu spectral avec un égaliseur paramétrique. Parascandolo *et al.* [83] appliquent d'autres transformations comme l'étirement temporel (simuler le fait qu'un son soit plus ou moins rapide), un décalage des frames et des mélanges de sons de même contexte. Uhlich *et al.* [114] ont choisi d'autres transformations comme un échange aléatoire entre les canaux gauche et droit, une mise à l'échelle de l'amplitude, un découpage aléatoire des sons ainsi que le mélange de différents instruments.

3.3 État de l'art en classification audio

La classification audio peut être divisée en deux catégories : la classification *monophonique* et *polyphonique*. La classification monophonique traite des cas où il y a une seule source présente à un instant t (classification multi-classe uni-label) alors que la classification polyphonique considère les cas où plus d'une source est présente à l'instant t (classification multi-classe multi-label). Une formalisation mathématique de ces définitions est donnée dans la partie 3.4. L'état de l'art des techniques de classification audio est pris sous l'angle des techniques utilisées dans chacun des cas monophonique et polyphonique.

Le cas de la classification monophonique est d'abord considéré. Les premières techniques ont utilisé les modèles de mélanges gaussiens (GMM) afin de modéliser les différentes classes de sons à retrouver. La règle de décision est alors le MAP. Li *et al.* [60] ont utilisé les attributs spectraux et cepstraux ainsi que les LPC (Linear Predictive Coefficients) pour classer des sons issus de programmes TV (principalement de la voix) : les MFCC donnaient les meilleures performances dans chaque expérience réalisée par rapport aux autres. Istrate *et al.* [44] ont réalisé un système de classification temps-réel pour monitorer la voix dans un contexte médical en utilisant les coefficients d'une décomposition en ondelettes et un système de classification en deux temps (classification en voix / son puis son normal / anormal). Clavel *et al.* [21] ont modélisé les MFCC et leurs dérivés avec des GMM pour la détection et classification de coups de feu. Une idée intéressante développée dans cet article était l'agrégation des probabilités *a posteriori* sur chaque frame en faisant le produit pour augmenter le taux de bonne détection. En effet cela permet d'avoir un plus grand horizon temporel (plutôt qu'une seule frame). Cette agrégation sera utilisée pour comparer les résultats dans la partie 3.7.

La classification audio se fait également avec les SVM, dans des cas généraux ou appliqués à la musique. La classification de genres musicaux a été traitée par Scaringella *et al.* [96] qui ont proposé un mélange d'experts de SVM utilisant les MFCC, avec des résultats assez variables suivant les genres (très bon pour le blues et le classique et mauvais pour les sons d'ambiance). Des approches «multi-résolutions» ont également été tentées en se basant sur plusieurs échelles temporelles pour calculer les attributs et une étape de fusion de données. Joder *et al.* [45] ont d'abord réalisé une sélection d'attributs par une discriminante de Fisher puis ont utilisé des

SVM pour la classification d'instruments. Pour la détection d'ambiance normale ou anormale, Lecomte *et al.* [55] ont construit des attributs à partir de banques de filtres en entrée d'un SVM. Kamruzzaman *et al.* [47] ont traité du cas de la classification de locuteurs en utilisant les MFCC et les SVM et ont obtenu d'assez bons résultats (jusque 95% de bonne classification avec 8 locuteurs).

Un aspect difficile de la classification audio est de prendre en compte les dépendances temporelles, ce qui est souvent fait avec les *chaînes de* Markov *cachées* (HMM). Parmi les premiers auteurs à considérer ces dernières, Rabiner *et al.* [88] ont proposé une HMM avec les MFCC pour faire de la reconnaissance vocale. Dans le cas des sons d'environnements, Heittola *et al.* [41] ont combiné une HMM avec des GMM utilisant des histogrammes pondérés d'évènements pour classer 61 évènements différents. Bietti *et al.* [13] ont généralisé les HMM à un contexte où deux variables servent à la chaîne de Markov (une variable cachée standard et une variable qui compte le temps passé dans l'évènement) pour la segmentation de notes de musiques.

Plus récemment, les techniques issues de l'apprentissage profond ont été considérées pour leur grande capacité de généralisation. Palaz *et al.* [81] ont utilisé le signal audio temporel en entrée d'un CNN à une dimension qui joue le rôle d'extracteur d'attributs et de classifieur. Piczak [85] a considéré les spectrogrammes audio comme des images puis a utilisé un CNN pour classer des sons d'environnements. Le CNN utilisé a une structure assez classique, représentée en Figure 3.1. Le prétraitement des données consiste à fixer la taille des sons à une taille commune à tous les sons (typiquement 10 secondes), puis à faire de l'augmentation de données (étirement/contraction temporelle, fréquentielle, décalage temporelle aléatoire). Ensuite le spectrogramme du son est calculé ainsi que sa dérivée suivant le temps. Les deux spectrogrammes sont ensuite considérés comme une image à deux canaux et constituent l'entrée du réseau de neurones. Cette méthode sera utilisée pour comparer les résultats dans la partie 3.7.

La classification polyphonique est plus complexe que la monophonique à cause de la présence simultanée de plusieurs classes. Une première approche considérée est l'utilisation de dictionnaires non-négatifs, avec l'utilisation de la NMF par exemple. Dikmen *et al.* [25] ont proposé d'apprendre simultanément un dictionnaire parcimonieux pour les spectrogrammes ainsi que les activations de classes, puis ont étendu cette méthode [68] dans le cas de données massives (avec réduction de dimension utilisant un algorithme *k-means*). Bisot *et al.* [15] ont proposé un apprentissage de NMF «guidé par la tâche» des spectrogrammes en échelle Mel.

L'apprentissage profond a été particulièrement utilisé dans le cas de la classification polyphonique. Adavanne *et al.* ont proposé plusieurs modèles pour la classification de signaux multicanaux en utilisant des attributs spectraux (énergies en échelle log-Mel, pitch) et spatiaux (différences de temps d'arrivée, TDOA) avec soit un CRNN (Convolutional Recurrent Neural Network) [2] ou un RNN avec LSTM [1]. Lee *et al.* [57] ont utilisé une méthode d'ensemble avec deux modèles (pour le morceau entier et un segment du morceau) qui sont spécialisés (prédiction du label ou des instants de détection), basés sur des couches convolutionnelles et le spectrogramme en échelle Mel. Espi *et al.* [32] ont utilisé une méthode multi-résolution : des spectrogrammes sont calculés à différentes résolutions et des CNN se chargent d'apprendre sur une résolution particulière, et enfin les résultats sont fusionnés. Dans le cas de la prédiction d'un instrument de musique prédominant, Han *et al.* [38] ont utilisé un CNN ainsi que le spectrogramme en échelle Mel. Parascandolo *et al.* [83] ont testé l'augmentation de données pour améliorer les performances d'un réseau BLSTM (Bidirectional LSTM). Çakir *et al.* ont d'abord étudié les meilleurs attributs issus d'un CNN initialisé par une échelle Mel [19], puis ont développé le réseau CRNN qui est la concaténation d'un CNN et d'un RNN, qui jouent respectivement le rôle d'extracteur d'attributs et de détecteur-classifieur respectivement [18]. La structure du CRNN est expliquée à la Figure 3.2. Ce réseau sera considéré comme méthode

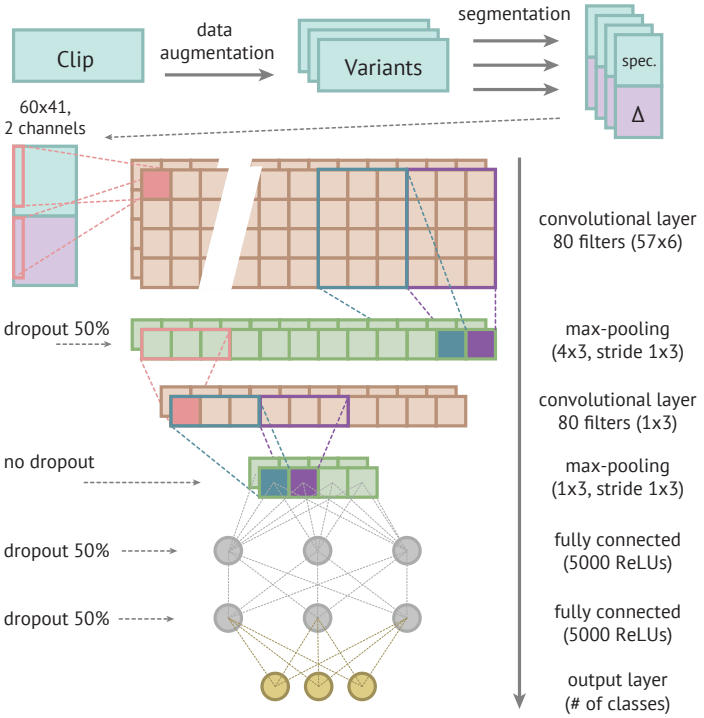


FIGURE 3.1 – Structure du CNN utilisé par PICZAK (extrait de [85]). Le spectrogramme ainsi que son delta (dérivée suivant le temps) sont calculés et sont considérés comme une image à deux canaux en entrée du CNN. Il comporte deux couches convolutionnelles avec max pooling ainsi que deux couches denses avant la couche de classification (softmax). Ce réseau de neurones sera considéré comme méthode concurrente pour la classification monophonique dans 3.7

concurrente pour la classification polyphonique dans 3.7. Bien souvent, la couche de sortie consiste en une couche dense dont les activations sont des sigmoides, et les valeurs sont seuillées pour obtenir un vecteur à entrées binaires représentant les activations des classes. Ce seuil est généralement fixé à 0,5.

Certains auteurs ont considéré des méthodes hybrides en combinant plusieurs algorithmes de bases. Hayashi *et al.* [40] ont modélisé les dépendances temporelles dans un réseau BLSTM avec des HSMM (Hidden Semi-Markov Model) pour contrôler plus finement la durée d'activation des fonctions mémoires. Benetos *et al.* [11] ont combiné la PLCA (pour modéliser les sources) et une HMM pour modéliser les transitions d'états. Enfin, Heittola *et al.* [42] ont d'abord réalisé une séparation de sources non supervisée avec NMF puis une classification supervisée.

De cet état de l'art on peut extraire plusieurs points non résolus que l'on tâchera d'étudier dans cette thèse :

1. Les auteurs considèrent des concaténations d'attributs et de méthodes spécifiquement à chaque cas d'utilisation mais qui ne sont pas souvent pertinents (au vu des résultats expérimentaux).
2. Les méthodes précédentes ne sont pas exploitables en temps-réel car elles utilisent soit trop de frames en entrée (vitesse faible) ou le coût computationnel est trop important (latence forte).
3. Les algorithmes de classification polyphoniques de ces auteurs ont plusieurs inconvénients tels que le seuillage de la sortie (d'où le choix d'un seuil optimal) ainsi que la nécessité d'entrainer le système sur un ensemble d'apprentissage comprenant des mélanges réels.

La prochain partie présente la méthode de classification monophonique développée dans la thèse, sur laquelle la méthode polyphonique s'appuiera.

3.4 Classification monophonique

3.4.1 Enoncé du problème

Nous allons présenter la méthode de classification audio en temps-réel développée dans cette thèse, appelée RARE (*Real-Time Audio Recognition Engine*). Le son est supposé contenir des évènements appartenant à certaines *classes de sons* (un avion, un coup de feu par exemple) que l'on cherche à prédire.

Définition 15 (Classification monophonique)**.** *L'objectif de classer à un instant t un son peut être écrit de la manière suivante :*

$$\widehat{z} = \underset{z}{\operatorname{argmax}}\, p\left(z \middle| f\left(\mathbf{x}^{temp}_{[t-\Delta T,t]}\right), t\right), \tag{3.1}$$

avec :

- $\widehat{z}$ une estimation du label $z \in \{1,...,K\}$ représentant la classe de son à un instant t (par exemple la classe $z = 1$ est composée de sons d'avions, $z = 2$ est composée de sons de coups de feu, *etc.*),
- $\mathbf{x}^{\text{temp}}$ est un son considéré comme un processus de longueur T et $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ est le son observé sur l'intervalle de temps $[t-\Delta T, t]$ (avec ΔT la durée d'observation),
- $f\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right)$ est une fonction qui calcule les attributs de $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ suivant les contraintes et objectifs décrits plus bas,

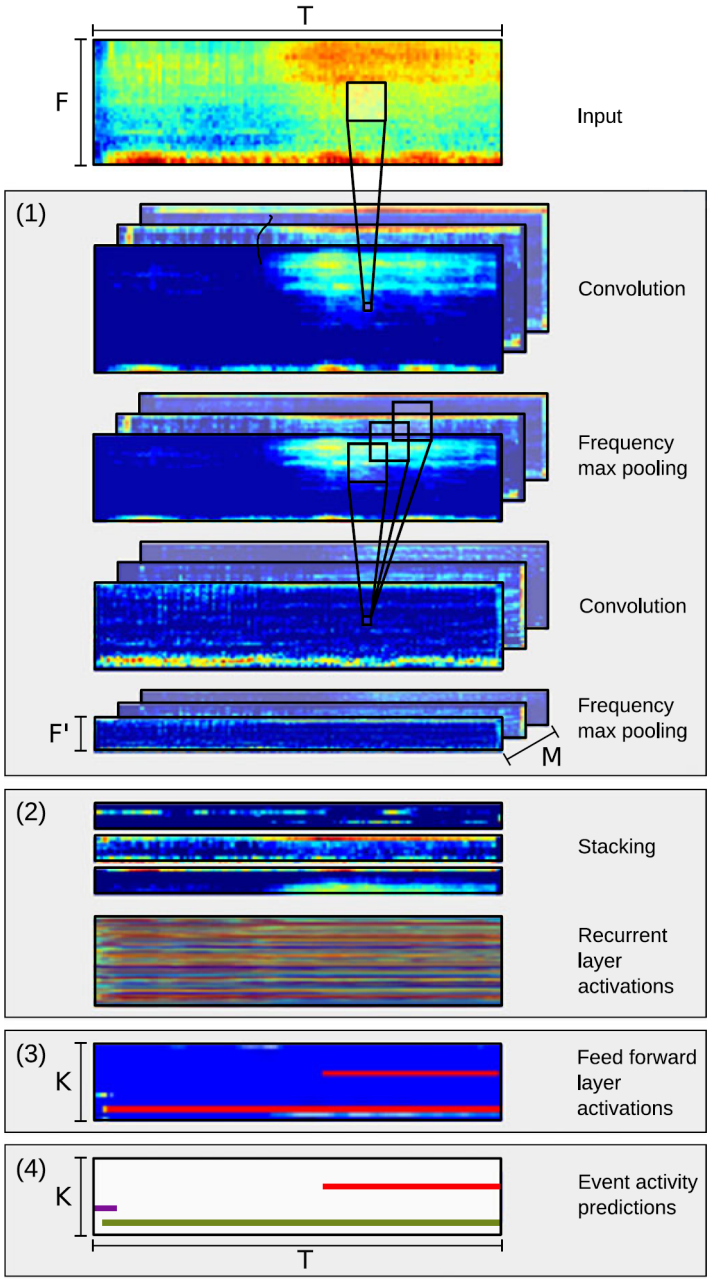


FIGURE 3.2 – Structure du CRNN utilisé par ÇAKIR dans [18]. L'idée est de combiner un CNN et un RNN pour tirer parti des meilleures propriétés de chacun. Le spectrogramme est passé dans un CNN pour extraire les attributs pertinents. Le RNN cherche à prédire les activations temporelles des classes en se basant sur la sortie du CNN. Ce réseau sera considéré comme méthode concurrente pour la classification polyphonique dans 3.7

.

— $p\left(z\middle|f\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right),t\right)$ est la probabilité de la classe z sachant les attributs extraits du son $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ au temps t.

La classification monophonique fait référence à la classification de frames *monophoniques*, qui est définie de la manière suivante :

Définition 16 (Frame monophonique)**.** *Dans une frame monophonique, le label z est supposé être le même sur l'intervalle de temps* $[t-\Delta T,t]$ *(continuité de z et au plus une source active à la fois).*

Plusieurs contraintes sont considérées pour la classification temps-réel. La première est le temps lié à l'acquisition de données qui contraint le temps t à être un multiple du *décalage temporel* δt, c'est-à-dire $t \in \{0, \delta t, 2\delta t, ...\}$. Ce décalage δt est important car la classification doit être réalisée à chaque multiple de δt, ce qui implique que le système doit être plus rapide que l'acquisition de données. La seconde contrainte est liée au *temps d'observation* ΔT qui correspond à la période d'observation du son, qui doit être fixée selon le temps de traitement voulu (voir partie correspondante sur les attributs). L'objectif de l'équation (3.1) étant posé, $f(\cdot)$ et $p(\cdot)$ sont choisis en conséquence dans les prochaines parties.

3.4.2 Conception des attributs

La fonction $f(\cdot)$ joue le rôle d'extracteur d'attributs de $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ pour réaliser la classification. En effet les signaux temporels ne sont pas adaptés pour la classification car ils ne sont pas assez discriminants : ils contiennent notamment l'information de phase et de volume pour lesquels nous souhaitons que notre méthode soit invariante. Le son est donc converti dans le domaine fréquentiel, et plus particulièrement le *spectre de puissance normalisé* est considéré. Ainsi en utilisant ce spectre de puissance normalisé nous avons bien invariance par rapport à la phase (puissance) et au volume (normalisation). De plus cette transformation sera particulièrement pertinente dans le cas polyphonique car elle conserve la propriété d'additivité sous hypothèse de signaux décorrélés (voir partie 3.5). Par conséquent, une définition possible de la fonction $f(\cdot)$ pourrait être $f_{\text{norm}} \circ f_{\text{TF}}(\cdot)$, avec f_{TF} la fonction qui calcule la Transformée de Fourier et f_{norm} la fonction qui normalise le spectre complexe de manière à obtenir le spectre de puissance normalisé (défini plus tard en équation 3.6).

Cependant de part l'aspect temps-réel, le signal $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ est trop long pour être utilisé tel quel. C'est pourquoi ces signaux sont découpés en *frames temporelles* :

$$\left(\mathbf{x}^{\text{temp}}_{1t}, ..., x^{\text{temp}}_{Nt}\right) = f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right), \tag{3.2}$$

avec $N = \lfloor(\Delta T - \Delta t)/\delta t\rfloor$ le nombre de frames qui peuvent être effectivement calculées pendant l'intervalle de temps $[t-\Delta T,t]$ et chaque frame $\mathbf{x}^{\text{temp}}_{\tau t} \in \mathbb{R}^{\Delta t}$ est définie par :

$$\mathbf{x}^{\text{temp}}_{\tau t} = \mathbf{w} \cdot \mathbf{x}^{\text{temp}}_{[t-\Delta T+(\tau-1)\delta t, t-\Delta T+(\tau-1)\delta t+\Delta t]},\ \tau = 1, ..., N. \tag{3.3}$$

avec $\mathbf{w}$ une fenêtre d'analyse (la fenêtre de Hanning par exemple). Une illustration est disponible en Figure 2.2 au chapitre 2. La frame $\mathbf{x}^{\text{temp}}_{\tau t}$ est un bout de $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ de taille Δt, chevauchant les frames précédentes par une durée $\tau\delta t$. La taille de frame Δt est fixée à une valeur faible pour avoir un calcul rapide de la Transformée de Fourier discrète. Des valeurs typiques sont par exemple 20ms, 50ms ou 100ms. En contrepartie la frame contiendra moins d'informations que $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$. Le choix de Δt implique donc un compromis entre temps de calcul et résolution du spectre : plus Δt est grande meilleure est la résolution mais plus grand est le temps de calcul. La valeur du décalage δt sera discutée durant la conception du modèle. Ainsi une collection

de plusieurs spectres de puissance normalisés est calculée plutôt qu'un seul grand spectre. On note :

$$(\mathbf{x}_{1t},...,\mathbf{x}_{Nt}) = f\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right), \tag{3.4}$$

$$= f_{\text{norm}} \circ f_{\text{TFD}} \circ f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right), \tag{3.5}$$

les spectres de puissance normalisés issus des frames temporelles (avec f_{TF} et f_{norm} qui opèrent sur chaque frame de manière indépendante), et où chaque spectre de puissance normalisé est calculé de la manière suivante :

$$\mathbf{x}_{\tau t} = \frac{\left|\mathbf{x}^{\text{freq}}_{\tau t}\right|^2}{\|\mathbf{x}^{\text{freq}}_{\tau t}\|^2}, \tag{3.6}$$

avec $\mathbf{x}^{\text{freq}}_{\tau t} \in \mathbb{C}^B$ la TFD de $\mathbf{x}^{\text{temp}}_{\tau t}$ dont on ne garde que $B \leq \Delta t$ bins fréquentiels, $|\cdot|$ le module élément par élément et $\|\cdot\|$ la norme-ℓ_2.

3.4.3 Conception du modèle

Le but de cette partie est de définir la distribution $p(\cdot)$. On considère un modèle génératif qui sera étendu dans le cas polyphonique et qui permet de n'apprendre que les classes monophoniques pour construire les classes polyphoniques. C'est un grand avantage comparé à de la modélisation prédictive ou à l'apprentissage multi-labels classique car il n'est pas nécessaire de posséder un ensemble de sons mélangés pour l'apprentissage. Le processus génératif de la collection de N spectres est :

— **Aléatoire** : Tirer un label : $z \sim \text{Mult}_K(\cdot; 1, \mathbf{p})$,
— **Aléatoire** : Tirer un son de longueur T de la classe z : $\mathbf{x}^{\text{temp}} \sim p_{\text{sound}}(\cdot|z)$,
— **Aléatoire** : Tirer un temps : $t \sim \mathcal{U}([0,T])$,
— **Déterministe** : Calculer les attributs : $(\mathbf{x}_{1t},...,\mathbf{x}_{Nt}) = f\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right)$,

avec $\mathcal{U}([a,b])$ la distribution uniforme sur l'intervalle $[a,b]$. Le processus génératif adopté a l'avantage de ne poser aucune hypothèse restrictive : la distribution qui génère les classes est la distribution multinomiale et la distribution qui génère les sons est supposée être une distribution générale sur les processus réels de taille T noté $p_{\text{sound}}(\cdot|z)$. Ce modèle génératif est complètement défini à $p_{\text{sound}}(\cdot|z)$ près, qui sera traité au moment de la modélisation conditionnelle.

On peut réécrire l'objectif de classification de l'équation 3.1 avec le théorème de Bayes :

$$p\left(z\middle|\mathbf{x}_{1t},...,\mathbf{x}_{Nt},t\right) \propto p\left(\mathbf{x}_{1t},...,\mathbf{x}_{Nt}\middle|z,t\right)p\left(z\middle|t\right). \tag{3.7}$$

Le processus génératif modélise la distribution jointe $p(\mathbf{x}_{1t},...,\mathbf{x}_{Nt},t,\mathbf{x}^{\text{temp}},z)$. Cependant seule la marginale $p\left(\mathbf{x}_{1t},...,\mathbf{x}_{Nt}\middle|z,t\right)$ est utile pour la classification. De part le processus génératif, on a directement que $p\left(z\middle|t\right) = p(z)$ car la classe est supposée être tirée avant le temps t.

En supposant que le décalage δt est suffisamment grand, deux frames consécutives sont considérées *indépendantes* (par définition du décalage δt, voir le Chapitre 2). En pratique pour $\delta t = \Delta t/2$ ou $\delta t = \Delta t/4$ (les valeurs considérées par la suite) il y a approximativement indépendance entre deux frames consécutives. La distribution de l'ensemble des spectres peut être alors approchée en utilisant une hypothèse d'indépendance conditionnelle par le produit suivant :

$$p\left(\mathbf{x}_{1t},...,\mathbf{x}_{Nt}\middle|z,t\right)=\prod_{\tau=1}^{N}p\left(\mathbf{x}_{\tau t}\middle|z,t\right). \tag{3.8}$$

L'équation (3.8) peut être vue comme une agrégation (comme en [5]) ou comme une hypothèse d'indépendance. Cette hypothèse est plutôt faible même si en pratique les signaux audio sont corrélés temporellement : par exemple cette hypothèse est pratique et déjà utilisée dans la littérature, comme dans [21].
Deux hypothèses supplémentaires sont faites. La première est une hypothèse de *stationarité en temps* : la distribution du spectre $\mathbf{x}_{\tau t}$ sachant la classe z ne dépend pas du temps t, ce qui se traduit par :

$$p\left(\mathbf{x}_{\tau t}\middle|z,t\right)=p\left(\mathbf{x}_{\tau t}\middle|z\right). \tag{3.9}$$

La seconde hypothèse est une hypothèse de *stationarité suivant les frames* : la distribution d'un spectre sachant la classe ne dépend pas du décalage τ (car la classe est la même sur l'intervalle de temps $[t-\Delta T,t]$), ce qui se traduit par :

$$p\left(\mathbf{x}_{\tau t}\middle|z\right)=p\left(\mathbf{x}_{\tau' t}\middle|z\right). \tag{3.10}$$

En utilisant les deux équations précédentes (3.9) et (3.10), on peut donc retirer les indices τ et t et écrire $\mathbf{x}_{\tau t}=\mathbf{x}$, ainsi que la distribution conditionnelle :

$$p\left(\mathbf{x}_{\tau t}\middle|z\right)=p\left(\mathbf{x}\middle|z\right). \tag{3.11}$$

Estimation du modèle

On suppose qu'on dispose d'un ensemble d'apprentissage $\left(\mathbf{x}_{(i)},z_{(i)}\right)_{i=1}^{n}$. La distribution (3.11) est estimée avec une estimation par noyaux de la forme :

$$\widehat{p}(\mathbf{x}|z)=\frac{1}{n_z}\sum_{\substack{i\\ z_{(i)}=z}}K\left(\mathbf{x},\mathbf{x}_{(i)}\right), \tag{3.12}$$

avec n_z le nombre d'échantillons dans la classe z, $K\left(\cdot,\mathbf{x}_{(i)}\right)$ un noyau à choisir. De part le modèle génératif précédent, nous avons également choisi une estimation de modèle très flexible via une estimation non paramétrique. Sachant la définition de $\mathbf{x}$ (vecteur qui somme à 1) on peut définir plusieurs noyaux :

- le noyau de Dirichlet, dont le paramètre est fixé de sorte que un spectre normalisé de l'apprentissage soit le mode de la distribution, c'est-à-dire $K\left(\cdot,\mathbf{x}_{(i)}\right)=\mathrm{Dir}\left(\cdot;\mathbf{x}_{(i)}+1\right)$. Ce noyau a l'avantage d'être simple et d'être cohérent avec la définition du spectre normalisé mais il n'est pas assez flexible. De plus, le calcul est lourd à cause de la fonction $\Gamma(\cdot)$ (pour rappel le calcul de la décision pour une frame doit être effectué en moins de temps que la durée de frame pour être considéré temps-réel).
- un mélange de Dirichlet de la forme :

$$K\left(\cdot,\mathbf{x}_{(i)}\right)=\sum_{m=1}^{M}\mathrm{Dir}\left(\cdot;\boldsymbol{\alpha}_{(i)m}\right), \tag{3.13}$$

avec M le nombre de composantes du mélange et $\boldsymbol{\alpha}_{(i)m}$ un paramètre qui dépend de $\mathbf{x}_{(i)}$. Ce dernier est plus flexible mais encore bien plus long à évaluer en temps réel.

D'autres noyaux pourraient également être considérés, mais le noyau choisi est le *noyau multinomial* : le spectre est quantifié de sorte qu'il devienne un vecteur d'entiers sommant à un facteur de quantification $q \in \mathbb{N}$. En effet ce noyau est facile à évaluer et permet de concevoir un compromis précision - temps de calcul comme décrit en partie 3.6 car seuls des produits scalaires sont calculés (la constante de normalisation est identique pour chaque noyau). Considérons $\mathbf{x}_{(i)}^{(q)} \in \mathbb{N}^B$ le vecteur d'entier le plus proche de $q\mathbf{x}_{(i)}$ est défini par :

$$\mathbf{x}_{(i)}^{(q)} = \underset{\mathbf{x}\in\mathbb{N}^B}{\operatorname{argmin}} \left\| q\mathbf{x}_{(i)} - \mathbf{x} \right\|. \tag{3.14}$$

Une solution à ce problème d'optimisation est de prendre le vecteur arrondi ou tronqué et n'apporte pas de charge de calcul supplémentaire. On définit le noyau d'approximation :

$$K^{(q)}\left(\cdot, \mathbf{x}_{(i)}^{(q)}\right) = \mathrm{Mult}_B\left(\cdot; q, \mathbf{p}_{(i)}^{(q)}\right), \tag{3.15}$$

dont le paramètre est défini par :

$$\mathbf{p}_i^{(q)} = \frac{1}{q}\mathbf{x}_{(i)}^{(q)}. \tag{3.16}$$

Théorème 3. *Le noyau d'approximation $K^{(q)}\left(\cdot, \mathbf{x}_{(i)}^{(q)}\right)$ converge vers $K\left(\cdot, \mathbf{x}_{(i)}\right)$ quand q tend vers l'infini et conserve les bonnes propriétés* [1] *du noyau initial.*

Démonstration. Considérons $\mathbf{X}_{(i)} \in \mathbb{R}^B$ un vecteur aléatoire qui somme à 1 lié au spectre $\mathbf{x}_{(i)}$ de l'ensemble d'apprentissage. $\mathbf{X}_{(i)}^{(q)}$ est défini comme le vecteur aléatoire entier le plus proche de $q\mathbf{X}_{(i)}$ par :

$$\mathbf{X}_{(i)}^{(q)} = \underset{\mathbf{X}\in\mathbb{N}^B}{\operatorname{argmin}} \left\| q\mathbf{X}_{(i)} - \mathbf{X} \right\|. \tag{3.17}$$

On a que $\frac{1}{q}\mathbf{X}_{(i)}^{(q)}$ converge en distribution vers $\mathbf{X}_{(i)}$ quand q tend vers l'infini. Ainsi le noyau d'approximation converge de manière ponctuelle vers le noyau original. □

Pour des valeurs assez grandes de q cette approximation sera correcte. En pratique, des valeurs de q proches de B sont suffisantes pour avoir une bonne approximation (voir partie 3.7).

Un autre noyau est le modèle de mélange gaussien pour des données par intervalle (noté bGMM) comme utilisé dans un de nos précédents articles [5]. En effet, on peut voir le spectre discrétisé comme un histogramme (données par intervalle) dont la distribution est un mélange gaussien. Celui-ci sera également considéré à titre de comparaison, tout comme le noyau de Dirichlet.

Les probabilités des classes sont estimées par maximum de vraisemblance :

$$\widehat{p_z} = \frac{n_z}{n}. \tag{3.18}$$

1. À savoir : fonction positive, normalisation et symétrie.

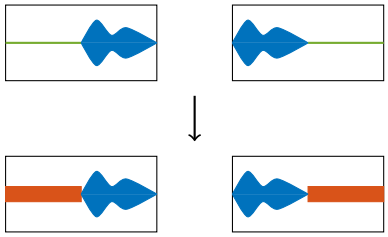


FIGURE 3.3 – Le phénomène suivant est illustré. En temps-réel, les sons (courbe pleine bleue) peuvent ne pas commencer au début de la frame (haut gauche) ou finir à la fin (haut droit). C'est pourquoi on ajoute (bas gauche et droit) du bruit blanc gaussien (rectangle rouge) pour remplir le «silence» (ligne verte).

Remarque. En pratique l'ensemble d'apprentissage est construit en utilisant le processus génératif avec quelques modifications. Un ensemble de sons est supposé déjà échantillonné suivant $p_{\text{sound}}(\cdot|z)$. On tire d'abord un label $z_{(i)}$ puis un son. L'étape de tirage d'un instant est différent mais imite le processus génératif : on considère plutôt tous les instants pour créer plusieurs frames. Le reste du processus est le même : les frames sont transformées en spectre de puissance normalisé. Le découpage implique que dans une frame donnée le son ne commence pas nécessairement au début ni ne finit forcément à la fin. Plutôt que d'ajouter du silence (qui contient une forme d'information), on ajoute du bruit blanc (qui ne contient pas d'information statistique) pour remplir le vide (voir FIGURE 3.3).

3.5 Classification polyphonique

3.5.1 Motivation

La classification polyphonique utilise l'apprentissage réalisé dans le cas monophonique. En décomposant d'une manière appropriée le spectre polyphonique, le modèle génératif polyphonique développé plus loin n'utilise que les modèles monophoniques et n'a pas besoin d'apprendre les mélanges de sons. Le cas d'un mélange de deux sources est considéré : cela signifie que la frame possède deux labels simultanément, notés $\mathbf{z} = (z_1, z_2) \in \{1, ..., K\}^2$ avec $z_1 \neq z_2$. L'objectif de classification polyphonique a une forme similaire à celui de la classification monophonique :

$$\widehat{\mathbf{z}} = \operatorname*{argmax}_{\mathbf{z}} p\left(\mathbf{z} \middle| f\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}, t\right)\right), \tag{3.19}$$

avec $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ le son polyphonique défini comme la somme de sons monophoniques $\mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]}$ et $\mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}$ correspondant aux classes z_1 et z_2 :

$$\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]} = \mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]} + \mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}. \tag{3.20}$$

Les mêmes attributs vont être utilisés car ils ont été conçus pour permettre l'extension au cas polyphonique, c'est-à-dire conserver l'additivité des signaux décorrélés.

3.5.2 Calcul des attributs polyphoniques

Cette partie montre que les attributs d'un son polyphonique se calculent à partir des attributs des sons monophoniques. L'étape de découpage étant linéaire, on a le résultat suivant (avec

l'addition élément par élément) :

$$f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{[t-\Delta T,t]}^{\text{temp}}\right) = f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{1[t-\Delta T,t]}^{\text{temp}}\right) + f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{2[t-\Delta T,t]}^{\text{temp}}\right). \tag{3.21}$$

Pour une frame τ cela donne :

$$\mathbf{x}_{\tau t}^{\text{temp}} = \mathbf{x}_{1\tau t}^{\text{temp}} + \mathbf{x}_{2\tau t}^{\text{temp}}. \tag{3.22}$$

Par linéarité de la TFCT, deux signaux sommés dans le domaine temporel le seront dans le domaine fréquentiel. On note $\mathbf{x}_{1\tau t}^{\text{freq}}$ et $\mathbf{x}_{2\tau t}^{\text{freq}}$ les spectres complexes des frames précédentes et $\mathbf{x}_{\tau t}^{\text{freq}}$ la somme définie par :

$$\mathbf{x}_{\tau t}^{\text{freq}} = \mathbf{x}_{1\tau t}^{\text{freq}} + \mathbf{x}_{2\tau t}^{\text{freq}}. \tag{3.23}$$

La modélisation introduite dans la partie monophonique utilise les spectres de puissance normalisés. L'hypothèse de *décorrélation* est utilisée pour permettre l'additivité des spectres de puissance :

$$\begin{aligned} \left|\mathbf{x}_{\tau t}^{\text{freq}}\right|^2 &= \left|\mathbf{x}_{1\tau t}^{\text{freq}} + \mathbf{x}_{2\tau t}^{\text{freq}}\right|^2, \\ &= \left|\mathbf{x}_{1\tau t}^{\text{freq}}\right|^2 + \left|\mathbf{x}_{2\tau t}^{\text{freq}}\right|^2. \end{aligned} \tag{3.24}$$

Le spectre de puissance normalisé associé à $\mathbf{x}_{\tau t}^{\text{freq}}$, noté $\mathbf{x}_{\tau t}$, est calculé avec :

$$\mathbf{x}_{\tau t} = \frac{\left|\mathbf{x}_{1\tau t}^{\text{freq}}\right|^2 + \left|\mathbf{x}_{2\tau t}^{\text{freq}}\right|^2}{\left\|\mathbf{x}_{1\tau t}^{\text{freq}}\right\|^2 + \left\|\mathbf{x}_{2\tau t}^{\text{freq}}\right\|^2}. \tag{3.25}$$

Les puissances des deux sources sont définies par $P_{1\tau t} = \left\|\mathbf{x}_{1\tau t}^{\text{freq}}\right\|^2$ et $P_{2\tau t} = \left\|\mathbf{x}_{2\tau t}^{\text{freq}}\right\|^2$. Quelques calculs permettent d'arriver au résultat suivant :

$$\mathbf{x}_{\tau t} = \mathbf{x}_{1\tau t}\frac{P_{1\tau t}}{P_{1\tau t} + P_{2\tau t}} + \mathbf{x}_{2\tau t}\frac{P_{2\tau t}}{P_{1\tau t} + P_{2\tau t}}, \tag{3.26}$$

avec $\mathbf{x}_{1\tau t}$ et $\mathbf{x}_{2\tau t}$ définis par l'équation (3.6). En définissant la proportion $\phi_{\tau t}$ par :

$$\phi_{\tau t} = \frac{P_{1\tau t}}{P_{1\tau t} + P_{2\tau t}} \tag{3.27}$$

finalement le résultat précédent peut se réécrire :

$$\mathbf{x}_{\tau t} = \phi_{\tau t}\mathbf{x}_{1\tau t} + (1 - \phi_{\tau t})\mathbf{x}_{2\tau t}, \tag{3.28}$$

ou en notation vectorisée :

$$f\left(\mathbf{x}_{[t-\Delta T,t]}^{\text{temp}}\right) = \boldsymbol{\phi}_t \cdot f\left(\mathbf{x}_{1[t-\Delta T,t]}^{\text{temp}}\right) + \left(\mathbf{1} - \boldsymbol{\phi}_t\right) \cdot f\left(\mathbf{x}_{2[t-\Delta T,t]}^{\text{temp}}\right), \tag{3.29}$$

avec $\mathbf{x} \cdot \mathbf{y}$ le produit élément par élément entre $\mathbf{x}$ et $\mathbf{y}$. Le fait de normaliser les spectres de puissances introduit ainsi une pondération $\boldsymbol{\phi}_t = (\phi_{1t}, ..., \phi_{Nt})$ entre les spectres monophoniques qui représentent le ratio de puissance entre les sources et leur somme (voir Figure 3.4). Ainsi les attributs polyphoniques sont construits à partir des attributs monophoniques.

Remarque. Nous supposons que le temps auquel les sons sont observés est le même, mais les sons individuels peuvent avoir deux temps de départ différents. La synchronisation sous-jacente

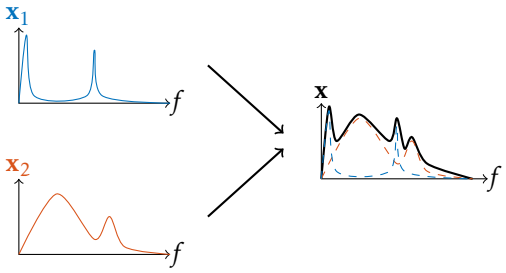


FIGURE 3.4 – Soient deux spectres de puissance normalisés $\mathbf{x}_1$ (courbe bleue en haut) et $\mathbf{x}_2$ (courbe rouge en bas) qui se mélangent. Le spectre résultant $\mathbf{x} = \phi\mathbf{x}_1 + (1-\phi)\mathbf{x}_2$ (courbe noire à droite) va dépendre du ratio entre les puissances de chaque source et leur somme représentée par ϕ.

des sons n'est pas explicitée par simplicité de présentation mais cela est bien évidemment pris en compte dans la méthode proposée.

3.5.3 Conception du modèle

Le modèle monophonique est étendu au cas polyphonique de la manière suivante :

— **Aléatoire** : Tirer sans remise deux labels : $z_1, z_2 \sim \text{Mult}_K(1, \mathbf{p})$,
— **Aléatoire** : Tirer deux sons différents : $\mathbf{x}_1^{\text{temp}} \sim p_{\text{sound}}(\cdot|z_1)$ et $\mathbf{x}_2^{\text{temp}} \sim p_{\text{sound}}(\cdot|z_2)$,
— **Aléatoire** : Tirer un temps : $t \sim \mathcal{U}([0, T])$,
— **Déterministe** : Calculer les attributs : $(\mathbf{x}_{1t}, ..., \mathbf{x}_{Nt}) = \phi_t \cdot f\left(\mathbf{x}_{1[t-\Delta T,t]}^{\text{temp}}\right) + \left(\mathbf{1} - \phi_t\right) \cdot f\left(\mathbf{x}_{2[t-\Delta T,t]}^{\text{temp}}\right)$.

Comme pour le cas monophonique on peut écrire l'objectif de classification avec le théorème de BAYES :

$$p(z_1, z_2|\mathbf{x}_{1t}, ..., \mathbf{x}_{Nt}, t) \propto p(\mathbf{x}_{1t}, ..., \mathbf{x}_{Nt}|z_1, z_2, t)\, p(z_1, z_2|t). \tag{3.30}$$

Le processus génératif modélise la distribution jointe $p\left(\mathbf{x}_{1t}, ..., \mathbf{x}_{Nt}, t, \mathbf{x}_1^{\text{temp}}, \mathbf{x}_2^{\text{temp}}, z_1, z_2\right)$, mais comme dans le cas monophonique seule la conditionnelle $p(\mathbf{x}_{1t}, ..., \mathbf{x}_{Nt}|z_1, z_2, t)$ est utile. De même grâce au processus génératif on a $p(z_1, z_2|t) = p(z_1, z_2)$. L'hypothèse d'indépendance des frames est encore valide ce qui conduit à l'approximation suivante :

$$p(\mathbf{x}_{1t}, ..., \mathbf{x}_{Nt}|z_1, z_2, t) \approx \prod_{\tau=1}^{N} p(\mathbf{x}_{\tau t}|z_1, z_2, t). \tag{3.31}$$

Les hypothèses de stationnarité (dans le temps Eq. 3.9 et sur les frames Eq. 3.10) développées en monophonique sont également valides :

$$p(\mathbf{x}_{\tau t}|z_1, z_2, t) = p(\mathbf{x}_{\tau t}|z_1, z_2). \tag{3.32}$$

On peut donc retirer les indices : $\mathbf{x}_{\tau t} = \mathbf{x}$ et la distribution s'écrit :

$$p(\mathbf{x}_{\tau t}|z_1, z_2, t) = p(\mathbf{x}|z_1, z_2). \tag{3.33}$$

3.5.4 Construction de l'ensemble d'apprentissage et estimation du modèle

L'avantage de la modélisation précédente est qu'elle permet de fabriquer un nouvel ensemble d'apprentissage pour les sons polyphoniques en utilisant uniquement les sons monophoniques. Cela évite d'avoir à enregistrer des sons polyphoniques et à les labelliser «à la main», ce qui peut être très long et coûteux [70]. La procédure est assez simple en utilisant le modèle génératif : on tire deux classes desquelles on tire deux sons. Chaque son est découpé en frames puis celles-ci sont transformées en spectres de puissance normalisés. Enfin les spectres polyphoniques sont calculés deux à deux en faisant la somme pondérée des spectres monophoniques avec la proportion correspondant $\phi_{(i)}$, de la forme :

$$\mathbf{x}_{(i)} = \phi_{(i)}\mathbf{x}_{1(i)} + \left(1 - \phi_{(i)}\right)\mathbf{x}_{2(i)}. \tag{3.34}$$

On obtient ainsi un ensemble d'apprentissage $\left(\mathbf{x}_{(i)}, \mathbf{z}_{(i)}\right)_{i=1}^{n}$ avec $\mathbf{z}_{(i)} = \left(z_{1(i)}, z_{2(i)}\right)$. On estime la distribution par des noyaux de la forme :

$$\widehat{p}(\mathbf{x}|\mathbf{z}) = \frac{1}{n_{z_1} n_{z_2}} \sum_{\substack{i \\ \mathbf{z}_{(i)}=\mathbf{z}}} K\left(\mathbf{x}, \mathbf{x}_{(i)}\right). \tag{3.35}$$

Tout comme la partie monophonique ce noyau va être approché par un noyau multinomial, qui aura lui aussi les bonnes propriétés de convergence. Cette méthode ne peut (théoriquement) classer que des sons polyphoniques dont les proportions sont les mêmes que celles de l'ensemble d'apprentissage. Nous verrons avec les expériences que la méthode fonctionne bien même avec des pondérations non vues dans l'apprentissage. Par indépendance, on a $p(z_1, z_2) = p(z_1)p(z_2)$ et l'estimation est directe (la même que pour le cas monophonique).

3.6 Réduire la complexité

Jusqu'à présent, la méthode de classification monophonique et polyphonique développée remplit le contrat initial de fonctionner sur des données en temps-réel :

— elle prend des spectres issus de frames en entrée ;
— elle utilise des noyaux multinomiaux rapides à évaluer.

Cependant le coût de calcul lié au nombre d'échantillons mis en jeu est potentiellement trop grand pour que le temps de calcul soit plus faible que la durée d'une frame. L'objectif de cette partie est de proposer une analyse de la complexité ainsi qu'une méthode pour diminuer le temps de calcul.

3.6.1 Évaluation de la complexité de la tâche de classification

L'équation (3.12) peut être écrite sous la forme :

$$\widehat{p}(\mathbf{x}|z) \propto \sum_{\substack{i \\ z_{(i)}=z}} \exp\left(\left(\mathbf{x}^{(q)}\right)^{\top} \log\left(\mathbf{p}_{(i)}^{(q)}\right)\right) \tag{3.36}$$

avec $\mathbf{x}^{\top}$ la transposée de $\mathbf{x}$ et $\log(\mathbf{x})$ le logarithme élément par élément de $\mathbf{x}$. La complexité de l'algorithme est de l'ordre de $\mathcal{O}(n)$ car il consiste à calculer un ensemble de produits scalaires. Même pour des petits ensembles d'apprentissage, le nombre de modèles peut être très grand (de l'ordre de 100k modèles), et donc le temps de calcul est plus grand que la durée d'une frame.

Même si le temps de calcul d'un produit scalaire était de 1µs cela reviendrait à un temps de calcul total de 1s pour une frame qui dure à peine 50ms. Cette règle de décision ne peut pas être changée (car elle vient du modèle génératif et de l'estimation par noyaux) pour réduire le temps de calcul, mais la réduction du nombre de modèles n va mécaniquement diminuer le temps de calcul.

3.6.2 Classification hiérarchique des modèles monophoniques

Une technique de réduction de modèle a déjà été considérée par Mesaros *et al.* [68] dans le cas de dictionnaires issus de la NMF pour la séparation de sources : les auteurs ont utilisé un *k-means* pour ne garder que les centroides comme leurs modèles réduits. Leurs résultats suggèrent que la précision d'un tel système n'est pas monotone avec le nombre de clusters considérés. Contrairement à ces auteurs, notre algorithme de réduction permet de contrôler le compromis précision - temps de calcul.

L'idée est de réaliser, dans chaque classe z, une classification hiérarchique des noyaux paramétrés par $\mathbf{p}_{(i)}^{(q)}$ pour $i|z_{(i)} = z$, et utiliser l'arbre de classification résultant pour créer des mélanges de noyaux suivant les clusters. La classification hiérarchique nécessite une distance entre les éléments à classer et un critère de lien. Nous avons considéré la distance de Hellinger [43] car les éléments sont des distributions de probabilités discrètes.

Définition 17 (Distance de Hellinger)**.** *La distance de* Hellinger *H entre deux distributions de probabilités P et Q s'écrit :*

$$H^2(P,Q) = \frac{1}{2}\int \left(\sqrt{\mathrm{d}P} - \sqrt{\mathrm{d}Q}\right)^2. \tag{3.37}$$

Dans le cas de distributions discrètes $\mathbf{p}$ *et* $\mathbf{q}$*, elle s'écrit :*

$$H^2(P,Q) = \frac{1}{2}\sum_i \left(\sqrt{p_i} - \sqrt{q_i}\right)^2. \tag{3.38}$$

Le critère de lien est celui de Ward [121] car il est très classique en classification hiérarchique. Celui-ci consiste à regrouper des éléments dans un groupe de sorte que la variance intragroupe après regroupement soit la plus faible possible et la variance intergroupe la plus grande possible.

Dans la classe z, une classification hiérarchique est réalisée avec un facteur de réduction par classe r_z (ou global r). Ce facteur est défini comme le nombre initial de noyaux n_z sur le nombre après réduction n'_z, soit : $r_z = n_z/n'_z$. Pour chaque cluster $\mathcal{C}_{i'} \in \{\mathcal{C}_1, ..., \mathcal{C}_{n'_z}\}$ résultant de la classification hiérarchique, $\mathcal{I}_{i'} = \left\{ i \,\middle|\, \mathbf{p}_{(i)}^{(q)} \in \mathcal{C}_{i'} \right\}$ représente l'ensemble des indices tels que $\mathbf{p}_{(i)}^{(q)}$ appartient au cluster $\mathcal{C}_{i'}$ ($i' = 1, ..., n'_z$). Un mélange $\overline{\mathbf{p}}_{(i')}^{(q)}$ des paramètres $\mathbf{p}_{(i)}^{(q)}$ dans le cluster $\mathcal{C}_{i'}$ est défini comme :

$$\overline{\mathbf{p}}_{(i')}^{(q)} = \frac{1}{\mathrm{card}\,(\mathcal{I}_{i'})} \sum_{i \in \mathcal{I}_{i'}} \mathbf{p}_{(i)}^{(q)} \tag{3.39}$$

avec $\mathrm{card}\,(A)$ le nombre d'éléments dans l'ensemble A. Le label associé au noyau de paramètre $\overline{\mathbf{p}}_{(i')}$ est noté $z_{(i')}$. Une illustration de ce procédé est disponible en Figure 3.5. Il sera montré en partie 3.7 que la méthode de réduction améliore les résultats jusqu'à une certaine valeur de réduction et suggère que la procédure permet en outre de limiter l'influence du surapprentissage le cas échéant.

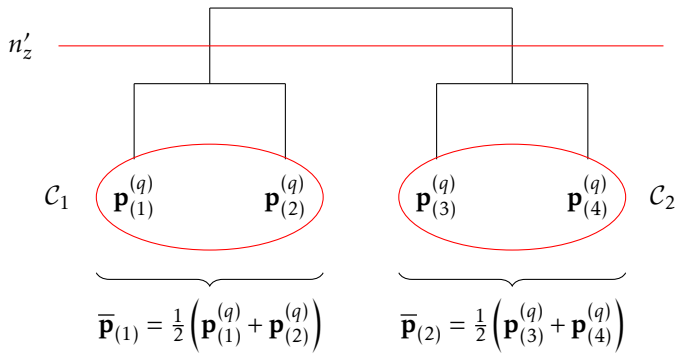


FIGURE 3.5 – Illustration de la réduction de complexité avec la classification hiérarchique. On considère que dans la classe z il y a $n_z = 4$ noyaux paramétrés par $\mathbf{p}^{(q)}_{(1)}, \ldots, \mathbf{p}^{(q)}_{(4)}$. La classification hiérarchique donne comme résultat un arbre représentant la proximité des noyaux. On choisit ensuite un nombre de noyaux résultant après la réduction n'_z (ici $n'_z = 2$) et l'arbre est «coupé» de manière à obtenir les clusters $\mathcal{C}_1, \mathcal{C}_2$. Finalement les noyaux sont groupés dans chaque cluster en prenant la moyenne pour obtenir $\overline{\mathbf{p}}_{(1)}$ et $\overline{\mathbf{p}}_{(2)}$.

3.6.3 Contrôle du compromis avec une procédure par seuillage

Un algorithme heuristique pour réduire les noyaux de manière plus efficace a été développé dans le cadre de cette thèse sur la base du précédent algorithme. Au lieu de réduire les classes avec le même facteur r, les classes sont réduites de manière indépendante en choisissant un facteur tel que la précision sur cette classe atteigne un seuil $t_{\text{precision}}$. On considère un facteur de réduction initial $r_z^{(0)}$, puis les noyaux sont réduits à partir de ce facteur et la précision par classe est calculée : si elle est au-dessus du seuil $t_{\text{precision}}$ la procédure s'arrête, sinon le facteur est ajusté de manière à augmenter la précision. Cette procédure est itérée jusqu'à ce que le seuil soit atteint dans chaque classe. Cette procédure peut être utilisée pour atteindre une précision donnée ou un temps de calcul donné. Un exemple de quelques itérations est donné dans le TABLEAU 3.10 de la partie 3.7.

Remarque : Réduction des modèles polyphoniques. Les expériences de la partie 3.7 montrent qu'un modèle très réduit peut être utilisé pour la classification monophonique sans perdre trop de précision. C'est pourquoi la classification polyphonique est basée sur ces noyaux réduits paramétrés par $\overline{\mathbf{p}}_{(i')}$ décrits dans la partie 3.6.2. Le but est de construire un ensemble de mélanges pertinents pour la classification polyphonique. La méthode consiste à calculer les mélanges deux à deux des noyaux réduits pour chaque classe, et si le temps de calcul utilisant ces noyaux est trop long, une étape de réduction de modèle peut être utilisée.

3.6.4 Cas particulier du noyau bGMM

Dans le cas du noyau bGMM (GMM estimé par intervalle), la procédure de réduction diffère légèrement. En effet, on peut calculer le paramètre du noyau $\mathbf{p}^{(q)}$ comme suit :

$$p_b^{(q)} = F_{(i)}(f_b + 1) - F_{(i)}(f_b), \tag{3.40}$$

avec $\mathbf{f} = ((b-1)F_e/\Delta t)_{b=1}^{B+1}$ le vecteur contenant les fréquences présentes dans le spectre et $F_{(i)}(\cdot)$ la fonction de répartition (c.d.f.) associée au GMM estimé sur le spectre $\mathbf{x}_{(i)}$.

Pour calculer la distance de Hellinger entre deux GMM de densité $g_1(\cdot)$ et $g_2(\cdot)$, on utilise l'approximation suivante :

$$H^2(g_1, g_2) = \frac{1}{2}\int_{f_1}^{f_{B+1}} \left(\sqrt{g_1(f)} - \sqrt{g_2(f)}\right)^2 \mathrm{d}f, \tag{3.41}$$

$$\approx \sum_{b=1}^{B}\left(\sqrt{\int_{f_b}^{f_b+1} g_1(f)\mathrm{d}f} - \sqrt{\int_{f_b}^{f_b+1} g_2(f)\mathrm{d}f}\right)^2, \tag{3.42}$$

$$= \sum_{b=1}^{B}\left(\sqrt{G_1(f_b+1) - G_1(f_b)} - \sqrt{G_2(f_b+1) - G_2(f_b)}\right)^2. \tag{3.43}$$

En pratique cette approximation est plutôt correcte et permet de calculer plus rapidement les distances. Si on souhaite avoir une estimation plus précise, on peut utiliser une méthode de Monte-Carlo pour échantillonner suivant les deux GMM et avoir une approximation de l'intégrale.

Une fois les distances calculées et l'arbre de classification créé, les paramètres $\mathbf{p}^{(q)}$ sont moyennés de la même manière que précédemment.

3.7 Expériences numériques sur données réelles.

Dans cette partie nous allons présenter les différentes bases de données permettant d'évaluer les performances de la méthode ainsi que les différents résultats obtenus sur la méthode de la thèse et les méthodes concurrentes.

3.7.1 Ensembles de données et métriques pour la classification audio temps-réel

Bases de données. Les ensembles de données communément utilisés pour évaluer les systèmes de classification monophonique et polyphonique sont les suivants :

- *A-Volute*. Cette base de sons monophoniques et polyphoniques comporte des sons de jeux vidéos répartis en 5 classes (alarme, détonation, moteur, bruit de pas, voix). C'est une base propriétaire qui appartient à A-Volute.
- *Battlefield*. Il s'agit de quelques enregistrements du jeu Battlefield utilisés pour la classification polyphonique. Cette base contient des mélanges de 3 classes différentes (voix, détonation et bruit de pas) avec au maximum 3 classes présentes en même temps.
- *ESC* (Environmental Sounds Classification dataset) [86]. Cette base de données monophoniques se compose de 50 classes de 2000 sons environnementaux, répartis suivant différentes catégories (sons d'animaux, paysage sonore naturel, bruits humains non parlés, sons domestiques d'intérieur, sons urbains d'extérieur). Une version réduite de cette base est ESC-10 et ne contient que 10 classes de sons parmi les 50.
- *TUT-SED* [70]. Cette base de données polyphoniques contient des scènes acoustiques issues de zones résidentielles ou de maisons. Il y a 6 classes différentes avec au maximum 4 classes présentes en même temps.

Un résumé des caractéristiques utiles de ces bases de données est disponible dans le Tableau 3.1.

Tableau 3.1 – Résumé des bases de données utilisées pour les expériences. La complexité polyphonique est le nombre maximum de classes mélangées simultanément : lorsque cette complexité est égale à 2, cela signifie qu'il y a au maximum 2 classes différentes mélangées simultanément.

Nom	Type	Nombre de classes	Complexité polyphonique
A-Volute	Monophonique	5	1
ESC-10	Monophonique	10	1
A-Volute	Polyphonique	5	2
Battlefield	Polyphonique	3	3
TUT-SED	Polyphonique	6	4

Métriques. Les métriques d'évaluation des systèmes de classification monophonique et polyphonique sont différentes. Dans le cas monophonique, on utilise le taux de bonne classification, c'est-à-dire le nombre de frames bien classées sur le nombre total de frames à classer. La matrice de confusion peut également être calculée pour avoir les métriques par classe. Les métriques polyphoniques peuvent être basées sur les évènements ou sur les frames audio [69]. Nous utiliserons celles basées sur les frames car elles sont plus adaptées au temps-réel. Deux métriques sont considérées : le score-F et le taux d'erreur. Le score-F nécessite les quantités suivantes :

— vrai positif (VP) : la référence et la sortie du système indique une classe active sur le segment,
— faux positif (FP) : la référence indique une classe inactive et la sortie du système indique une classe active,
— faux négatif (FN) : la référence indique une classe active et la sortie du système indique une classe inactive.

A partir de ces quantités on définit la précision P et le rappel R :

$$\mathrm{P} = \frac{\mathrm{VP}}{\mathrm{VP}+\mathrm{FP}}, \tag{3.44}$$

$$\mathrm{R} = \frac{\mathrm{VP}}{\mathrm{VP}+\mathrm{FN}}. \tag{3.45}$$

Le score-F (F1) se calcule de la manière suivante :

$$\mathrm{F1} = \frac{2\mathrm{P}\cdot\mathrm{R}}{\mathrm{P}+\mathrm{R}}. \tag{3.46}$$

Le taux d'erreur (E.R.) utilise quatre quantités qui dépendent de la frame τ :

— le nombre d'erreurs de substitution $\mathrm{S}(\tau) = \min(\mathrm{FN}(\tau), \mathrm{FP}(\tau))$: c'est le nombre d'évènements de référence pour lesquels un évènement correct n'a pas été prédit mais autre chose oui,
— le nombre d'insertions $\mathrm{I}(\tau) = \max(0, \mathrm{FP}(\tau) - \mathrm{FN}(\tau))$: le nombre d'évènements prédits qui ne sont pas corrects,
— le nombre d'effacements $\mathrm{D}(\tau) = \max(0, \mathrm{FN}(\tau) - \mathrm{FP}(\tau))$: le nombre d'évènements de référence qui n'ont pas été correctement identifiés,
— Le nombre d'évènements de référence actifs $\mathrm{N}(\tau)$.

Le taux d'erreur se calcule de la manière suivante :

$$\text{E.R.} = \frac{\sum_\tau \text{S}(\tau) + \sum_\tau \text{D}(\tau) + \sum_\tau \text{I}(\tau)}{\sum_\tau \text{N}(\tau)}. \tag{3.47}$$

3.7.2 Préparation des données

Les systèmes de classification audio reposent sur un ensemble de sons qui servent à entrainer ce système. Dans le but de pouvoir étudier le pouvoir de généralisation d'un système d'apprentissage, l'ensemble des données est divisé en un ensemble d'apprentissage et un ensemble de test (et parfois un ensemble de validation). Habituellement dans le cadre de la classification audio, ces ensembles sont créés à partir des sons. C'est-à-dire, si on dispose de n sons dans la classe z, et que l'on souhaite faire une division à 50% par exemple, on va prendre $n/2$ sons que l'on va mettre dans l'ensemble d'apprentissage et $n/2$ dans l'ensemble de test. Ensuite, chaque ensemble est traité de manière séparée : découpage en frames temporelles, calcul des spectres sonores, extraction des attributs, *etc*. De plus, les sons sont tous supposés de même longueur. S'il se trouve qu'un son n'a pas la bonne longueur, deux options sont envisagées : soit du silence est ajouté au début et/ou à la fin du son, soit la source d'intérêt est noyée dans un contexte audio (qui n'est pas nécessairement pertinent pour la classification).

Une préparation de données alternative a été proposée pour se rapprocher de notre cas d'utilisation qui est la classification temps-réel (à la frame). Celle-ci consiste à diviser l'ensemble de données sur les frames et non sur les sons. Cette préparation est censée améliorer les résultats puisque la donnée de départ étant la frame, il semble a priori judicieux de réaliser le découpage de l'ensemble d'apprentissage suivant les frames. Néanmoins nous verrons avec quelques expériences que ce découpage amène un sur-apprentissage pour toutes les méthodes. En effet, de part ce découpage, des frames adjacentes pourraient se retrouver dans les deux ensembles (apprentissage et test) et ainsi la ressemblance entre les deux serait trop proche.

3.7.3 Design des expériences

Réglage des paramètres. Tous les sons sont ré-échantillonnés à F_e =44,1kHz et centrés. La taille de fenêtre prend les valeurs suivantes $\Delta t = 512, 1024, 2048$ échantillons (11,6ms, 23,2ms and 46,4ms respectivement) et le décalage temporel est fixé à $\delta t = 512$ échantillons (11,6ms). Le nombre de fréquences gardées est $B = \Delta t/2 + 1$ (la première moitié du spectre plus la fréquence de Nyquist). La quantification du spectre prend les valeurs suivantes $q = 10, 10^2, B, 10^4, 10^5$. Les ensembles d'apprentissage sont divisés suivant le schéma v-fold pour la validation croisée (avec $v = 5$). Le nombre de frames à classer est fixé à $N = 40$, ce qui correspond à environ 500ms de données audio pour prendre la décision. On va chercher à prédire les sons de test avec le découpage suivant les frames puis les sons de test avec le découpage suivant les sons. On fait également varier le type de noyau : multinomial, Dirichlet et bGMM.

Méthodes concurrentes monophoniques. La méthode proposée pour la classification monophonique est comparée à plusieurs systèmes de référence. Le premier est un GMM avec 10, 20, 30 et 40 composants par classe utilisant 20 MFCC et leurs première et seconde dérivées, inspiré par [21]. Le second est également un GMM avec 20 et 30 composants par classe mais utilisant le spectre de puissance normalisé (NPS) comme descripteur. La troisième est une SVM multi-classes entraînée à l'aide de la fonction de coût ECOC (*Error Correction Output Coding*) avec un apprentissage en OVO. Le quatrième est un Deep Convolutional Neural Network (DCNN) utilisant des spectrogrammes log-Mel inspirés de [85], entrainé sur 50 itérations, avec une taille

Tableau 3.2 – Tâche de classification monophonique : schéma v-fold sur les frames. Précision moyenne de la classification par validation croisée et écart type (en %) en fonction du facteur de quantification q de la méthode RARE sur les bases A-Volute et ESC-10. Le facteur de quantification q varie dans l'ensemble $\left\{10^1, 10^2, B, 10^4, 10^5\right\}$. La valeur de la largeur de frame est $\Delta t = 2048$.

q	A-Volute	ESC-10
10^1	96,6 (0,40)	96,1 (0,4)
10^2	98,8 (0,01)	99,4 (0,1)
B	98,8 (0,01)	99,4 (0,1)
10^4	98,8 (0,01)	99,4 (0,1)
10^5	98,8 (0,01)	99,4 (0,1)

de batch de 64 éléments et un arrêt prématuré de l'apprentissage si la fonction de coût sur l'ensemble de validation reste stationnaire. Un modèle de mélange de Dirichlet (DMM) [64] est considéré de sorte qu'une classe est modélisée par un mélange de distribution de Dirichlet, et le nombre de composants par classe est réglé à 20 et 30. Des tests d'écoute ont été réalisés sur les différentes bases de données monophoniques : les résultats pour la base ESC-10 sont déjà disponibles dans [86]. Pour la base A-Volute une expérience a été réalisée sur 27 personnes qui devaient classer 50 sons d'une durée de 0,5 secondes choisis aléatoirement sur 5 classes.

Méthodes concurrentes polyphoniques. La méthode de classification polyphonique a également été comparée à une méthode basée sur un réseau de neurones, appelée CRNN, inspirée de [18], le réseau étant entrainé sur 100 itérations. Il utilise le spectrogramme log-Mel pour alimenter un réseau de neurones qui consiste en des couches convolutionnelles (pour l'extraction d'attributs) suivies de couches récurrentes (pour la détection temporelle) et terminées par une couche dense qui effectue la classification. Comme cette méthode utilise habituellement un ensemble de données avec des sons déjà mélangés, nous avons créé des mélanges artificiels sur l'ensemble de données A-Volute pour entraîner le réseau.

3.7.4 Résultats de la classification monophonique

Reconnaissance sur l'ensemble test découpé sur les frames

L'influence de la valeur de la quantification est montrée dans le Tableau 3.2. Pour un facteur de quantification plus grand que 100, la précision dépasse 96% pour les deux bases considérées. La méthode donne déjà des résultats très bons (quasiment parfait) sur les deux ensembles de données.

L'influence de la taille de fenêtre est disponible dans le Tableau 3.3. Les résultats augmentent avec la taille de fenêtre pour toutes les méthodes. Néanmoins, on remarque d'ors et déjà l'effet du sur-apprentissage dû au schéma v-fold sur les frames : en effet les résultats sont extrêmement bons (quasiment 100%) et la variance entre les folds est très faible (moins de 1%). Cela corrobore l'analyse que l'on avait faite précédemment. Ce schéma de v-fold est donc abandonné dans la suite de cette partie.

Tableau 3.3 – Tâche de classification monophonique : schéma v-fold sur les frames. Précision moyenne de la classification par validation croisée et écart type (en %) pour différentes valeurs de Δt sur l'ensemble de données A-Volute et pour les différentes méthodes. Les résultats sont obtenus en utilisant le modèle complet de la méthode proposée (non réduit).

Méthode		Param.	$\Delta t = 512$	$\Delta t = 1024$	$\Delta t = 2048$
RARE	Multinomial	$r = 1$	95,1 (0,5)	97,5 (0,3)	98,8 (0,1)
RARE	Dirichlet	$r = 1$	20,2 (0,2)	20,3 (0,1)	20,4 (0,1)
RARE	bGMM	$r = 1$	93,1 (0,7)	96,7 (0,4)	98,3 (0,3)
GMM [21]	MFCC	10 comp.	95,6 (2,7)	97,7 (0,3)	91,7 (1,3)

Tableau 3.4 – Tâche de classification monophonique : schéma v-fold sur les sons. Précision moyenne de la classification par validation croisée et écart type (en %) en fonction du facteur de quantification q de la méthode RARE sur les bases A-Volute et ESC-10. Le facteur de quantification q varie dans l'ensemble $\left\{10^1, 10^2, B, 10^4, 10^5\right\}$.

q	A-Volute	ESC-10
10^1	80,2 (5,1)	54,9 (4,5)
10^2	82,4 (5,4)	64,6 (3,0)
B	82,4 (5,4)	64,7 (2,9)
10^4	82,5 (5,4)	64,7 (2,9)
10^5	82,5 (5,4)	64,7 (2,9)

Reconnaissance sur l'ensemble test découpé sur les sons

L'influence de la valeur de la quantification est montrée dans le Tableau 3.4. Plus le facteur de quantification augmente, meilleurs sont les résultats (80,2% pour $q = 10^1$, 82,4% pour $q = 10^5$). Ce résultat est logique étant donné que le facteur de quantification contrôle l'approximation du noyau multinomial, et nous avons démontré qu'asymptotiquement ce noyau converge vers le noyau original. De plus, le spectre quantifié étant considéré comme un histogramme par la loi multinomial, il est normal qu'en deça d'une quantification $q = B$ la précision diminue car B est le nombre minimal d'observations qu'il faut dans un vecteur de taille B pour estimer une loi uniforme discrète. Il est remarquable que la base de données ESC-10 soit plus influencée par une faible valeur de q que la base A-Volute. Cela est sans doute dû à la diversité des sons présents dans ESC-10, et également le fait que ce sont des sons réels et non synthétisés comme dans la base A-Volute.

Méthode proposée sans réduction. Les résultats du Tableau 3.5 montrent l'influence de la longueur de frame Δt pour différentes méthodes sur le jeu de données A-Volute. La méthode proposée n'est pas significativement influencée par la valeur de Δt avec le noyau multinomial, contrairement aux autres méthodes. On remarque que le noyau de Dirichlet n'est pas pertinent car pas assez flexible dans sa modélisation (résultats beaucoup plus faibles que le noyau multinomial), alors que le noyau bGMM a des résultats similaires au noyau multinomial. La méthode GMM avec comme attributs le NPS, la SVM et le DCNN ont de meilleurs résultats avec des Δt plus grands. Cependant, les résultats sont plus équilibrés pour la méthode GMM avec comme attribut MFCC et un nombre différent de composantes : plus le nombre de composantes est élevé,

Tableau 3.5 – Tâche de classification monophonique : schéma v-fold sur les sons. Précision moyenne de la classification par validation croisée et écart type (en %) pour différentes valeurs de Δt sur l'ensemble de données A-Volute et pour les différentes méthodes. Les résultats sont obtenus en utilisant le modèle complet de la méthode RARE (non réduit).

Méthode	Attributs	Param.	$\Delta t = 512$	$\Delta t = 1024$	$\Delta t = 2048$
RARE	Multinomial	$r = 1$	81,7 (6,7)	81,9 (7,0)	82,5 (6,0)
RARE	Dirichlet	$r = 1$	20,7 (0,8)	21,6 (2,1)	21,5 (1,2)
RARE	bGMM	$r = 1$	80,4 (6,7)	81,1 (7,1)	82,8 (6,4)
GMM [21]	MFCC	10 comp.	87,2 (4,5)	87,0 (4,0)	88,1 (7,0)
		20 comp.	85,5 (4,7)	86,3 (4,5)	86,6 (8,5)
		30 comp.	84,0 (6,2)	83,5 (4,0)	84,5 (7,1)
		40 comp.	80,3 (4,6)	83,6 (1,9)	74,4 (2,4)
GMM [21]	NPS	20 comp.	70,8 (7,5)	75,3 (7,6)	77,2 (7,3)
		30 comp.	70,3 (4,6)	75,3 (8,2)	88,0 (7,5)
DMM	NPS	20 comp.	83,1 (3,5)	84,2 (6,1)	83,9 (3,9)
		30 comp.	83,5 (3,8)	84,4 (6,7)	84,7 (4,7)
SVM	NPS	-	23,7 (1,5)	27,2 (5,5)	36,9 (9,2)
DCNN [85]	MFCC	-	58,5 (12,8)	60,0 (14,5)	61,6 (16,0)
Humain			-	-	91,7

moins la méthode est efficace. Ce phénomène est vraisemblablement lié à un surapprentissage des GMM pour un nombre de composantes élevé. En raison de la dimensionnalité du NPS par rapport au MFCC, le GMM est moins performant avec le NPS qu'avec le MFCC. Ainsi, la méthode présentant les meilleurs résultats est le GMM avec MFCC à 10 composantes, et on remarque surtout les faibles résultats du DNN.

Les Tableaux 3.6 et 3.7 résument les résultats pour les ensembles de données A-Volute et ESC-10 respectivement pour : les différentes méthodes (RARE, GMM, SVM et DCNN), les différents attributs (NPS ou MFCC) et le paramètre de réglage correspondant (si nécessaire). Les résultats sont la précision (en %), le temps d'entraînement (en secondes) et le temps de test (en millisecondes). Dans le cas du GMM, l'hyperparamètre à régler est le nombre de composants du mélange. Le facteur de réduction r dans la méthode proposée n'est pas vraiment un hyperparamètre puisqu'il contrôle le compromis entre la précision et le temps de calcul. En terme de précision, la meilleure méthode est le GMM avec un nombre relativement faible de composants mais RARE est la deuxième méthode et obtient de bons résultats par rapport aux autres méthodes, en particulier par rapport au DCNN et à la SVM. La précision sur l'ensemble de données ESC-10 pour la méthode proposée diminuent parce qu'une classe (sur les 10) n'est pas bien reconnue, mais dans l'ensemble les résultats sont semblables à ceux de l'ensemble de données A-Volute : ce phénomène est visible dans les matrices de confusion 3.8 et 3.9.

Méthode proposée avec réduction. L'algorithme de réduction du modèle semble être très efficace car le nombre de noyaux peut être réduit d'un facteur 400 sans perdre trop de précision, mais il diminue considérablement le temps de test (voir Tableaux 3.6 et 3.7). Pour un facteur de réduction de $r = 10$, la méthode proposée peut être utilisée en temps réel car le temps de test est inférieur à 50ms (ce qui correspond à la longueur de frame). De plus, pour les facteurs de

Tableau 3.6 – Tâche de classification monophonique : schéma v-fold sur les sons. Résumé des résultats de l'ensemble de données A-Volute en termes de précision, de temps d'apprentissage (en secondes) et de temps de test par image (en millisecondes) pour différentes méthodes.

Méthode	Attributs	Param.	Précision (%)	Apprentissage	Test
RARE	Multinomial	$r = 1$	82,5 (6,0)	$1,0 \times 10^3$	$2,4 \times 10^2$
		$r = 2$	82,0 (6,2)		$1,3 \times 10^2$
		$r = 10$	83,1 (5,6)		$2,5 \times 10^1$
		$r = 50$	81,1 (5,3)		$5,6 \times 10^0$
		$r = 100$	79,9 (5,7)		$2,7 \times 10^0$
		$r = 400$	77,7 (5,3)		$0,8 \times 10^0$
RARE	Dirichlet	$r = 1$	21,5 (1,2)	$1,0 \times 10^3$	$3,7 \times 10^2$
		$r = 2$	21,2 (0,8)		$1,9 \times 10^2$
		$r = 10$	21,2 (1,1)		$3,8 \times 10^1$
		$r = 50$	21,3 (1,2)		$8,0 \times 10^0$
		$r = 100$	21,5 (1,6)		$4,1 \times 10^0$
		$r = 400$	21,5 (1,2)		$1,4 \times 10^0$
RARE	bGMM	$r = 1$	82,8 (6,4)	$3,6 \times 10^4$	$2,4 \times 10^2$
		$r = 2$	82,8 (7,0)		$1,3 \times 10^2$
		$r = 10$	82,7 (6,4)		$2,5 \times 10^1$
		$r = 50$	80,3 (6,5)		$5,6 \times 10^0$
		$r = 100$	79,4 (5,9)		$2,7 \times 10^0$
		$r = 400$	76,3 (7,0)		$0,8 \times 10^0$
GMM [21]	MFCC	10 comp.	88,1 (7,0)	$8,9 \times 10^1$	$3,0 \times 10^0$
		20 comp.	86,6 (8,5)	$1,3 \times 10^2$	$3,0 \times 10^0$
		30 comp.	84,5 (7,1)	$1,9 \times 10^2$	$4,1 \times 10^0$
		40 comp.	74,4 (2,4)	$2,4 \times 10^2$	$4,7 \times 10^0$
GMM [21]	NPS	20 comp.	77,2 (7,3)	$7,3 \times 10^2$	$2,8 \times 10^3$
		30 comp.	88,0 (7,5)	$1,3 \times 10^2$	$4,0 \times 10^3$
DMM	NPS	20 comp.	83,9 (3,9)	$1,5 \times 10^4$	$4,3 \times 10^0$
		30 comp.	84,7 (4,7)	$2,6 \times 10^4$	$6,3 \times 10^0$
SVM	NPS	-	36,9 (9,2)	$1,2 \times 10^4$	$2,9 \times 10^1$
DCNN [85]	MFCC	-	61,6 (16,0)	$1,4 \times 10^3$	$3,1 \times 10^0$
CRNN [18]	MFCC	-	19,2 (1,6)	$4,0 \times 10^2$	$2,9 \times 10^1$
Humain	-	-	91,7	-	-

Tableau 3.7 – Tâche de classification monophonique : schéma v-fold sur les sons. Résumé des résultats de l'ensemble de données ESC-10 en termes de précision, de temps d'apprentissage (en secondes) et de temps d'essai par trame (en millisecondes) pour différentes méthodes.

Méthode	Attributs	Param.	Précision (%)	Apprentissage	Test
RARE	Multinomial	$r = 1$	64,7 (3,2)	$7,9 \times 10^2$	$3,7 \times 10^2$
		$r = 2$	63,9 (4,4)		$1,8 \times 10^2$
		$r = 10$	67,5 (4,4)		$3,7 \times 10^1$
		$r = 50$	67,8 (3,9)		$7,9 \times 10^0$
		$r = 100$	67,1 (5,1)		$4,0 \times 10^0$
		$r = 400$	61,7 (3,8)		$1,2 \times 10^0$
RARE	Dirichlet	$r = 1$	30,1 (3,1)	$7,9 \times 10^2$	$5,1 \times 10^2$
		$r = 2$	30,3 (3,8)		$2,5 \times 10^2$
		$r = 10$	30,5 (3,7)		$5,1 \times 10^1$
		$r = 50$	28,0 (3,0)		$1,0 \times 10^1$
		$r = 100$	28,2 (3,2)		$5,5 \times 10^0$
		$r = 400$	29,8 (3,5)		$1,6 \times 10^0$
RARE	bGMM	$r = 1$	71,9 (4,9)	$3,5 \times 10^4$	$3,7 \times 10^2$
		$r = 2$	70,6 (4,7)		$1,8 \times 10^2$
		$r = 10$	69,3 (4,6)		$3,7 \times 10^1$
		$r = 50$	67,1 (4,8)		$8,0 \times 10^0$
		$r = 100$	65,5 (4,9)		$5,9 \times 10^0$
		$r = 400$	61,5 (5,8)		$1,6 \times 10^0$
GMM [21]	MFCC	10 comp.	78,9 (4,1)	$1,5 \times 10^2$	$2,5 \times 10^0$
		20 comp.	78,1 (4,0)	$2,2 \times 10^2$	$3,0 \times 10^0$
		30 comp.	78,3 (3,5)	$3,0 \times 10^2$	$3,5 \times 10^0$
		40 comp.	77,4 (4,0)	$3,6 \times 10^2$	$3,7 \times 10^0$
GMM [21]	NPS	20 comp.	56,7 (3,5)	$1,2 \times 10^3$	$5,7 \times 10^3$
		30 comp.	56,1 (4,2)	$2,0 \times 10^3$	$8,0 \times 10^3$
DMM	NPS	20 comp.	70,4 (5,3)	$2,7 \times 4$	$1,0 \times 10^1$
		30 comp.	71,5 (4,6)	$4,1 \times 10^4$	$1,0 \times 10^1$
SVM	NPS	-	46,3 (4,4)	$2,2 \times 10^4$	$4,7 \times 10^1$
DCNN [85]	MFCC	-	40,7 (6,0)	$2,1 \times 10^3$	$1,3 \times 10^0$
CRNN [18]	MFCC	-	10,0 (0,0)	$6,0 \times 10^2$	$2,6 \times 10^1$
Human	-	-	95,7	-	-

Tableau 3.8 – Matrice de confusion pour la base ESC-10 avec RARE. Les classes sont : tronçonneuse, tic d'horloge, crépitement de feu, pleurs de bébé, aboiement, hélicoptère, pluie, coq, bruit de vague et éternuements.

				Classes prédites						
Classes théoriques	**70,1**	0,2	1,5	0,2	0,2	2,4	13,9	0	11,4	0
	5,8	**7,0**	25,3	22,1	2,2	6,3	13,6	0	12,1	5,6
	0	0,2	**92,9**	1,4	0	5,2	0,2	0	0,1	0
	0	1,2	1,1	**91,6**	2,6	0	0,5	0,8	1,4	0,8
	0,4	0	0	12,2	**80,1**	0,8	0,1	0,9	4,6	1,0
	1,3	0,2	9,5	0	0,1	**69,9**	5,7	0	13,3	0
	1,9	0	8,6	0,7	0	1,6	**68,6**	0,3	18,3	0
	3,2	0,6	0,2	19,3	11,7	0	0,3	**64,1**	0,1	0,6
	0,2	0,2	2,1	0	0,3	1,4	5,4	0,1	**90,3**	0
	6,0	6,0	6,5	17,9	8,2	0	5,0	2,3	4,8	**43,3**

Tableau 3.9 – Matrice de confusion pour ESC-10 avec le GMM avec MFCC et 40 composantes. Les classes sont : tronçonneuse, tic d'horloge, crépitement de feu, pleurs de bébé, aboiement, hélicoptère, pluie, coq, bruit de vague et éternuements.

				Classes prédites						
Classes théoriques	**78,1**	0,4	0,5	4,7	1,7	0	1,0	0	5,6	8,2
	0,2	**76,0**	4, 6	18,9	0	0	0	0	0	0,3
	0,2	2,1	**87,5**	1,4	1,4	0	3,1	0	0	4,3
	0	0	0,6	**97,4**	1,8	0	0	0	0	0,3
	0,3	0,8	0	17,3	**77,8**	0	0	2,3	0	1,4
	2,7	2,2	5,1	0,9	2,1	**71,1**	1,4	0,3	10,5	3,9
	6,3	0	7,7	0,3	0,1	2,1	**66,3**	0	14,1	3,1
	0	0	0,2	18,6	1,4	0	0	**79,0**	0	0,8
	3,1	0	0	2,3	0	0,3	3,6	0,1	**90,5**	0,3
	0	0	1,3	44,9	2,1	0	0	1,2	0,1	**50,3**

Tableau 3.10 – Tâche de classification monophonique : schéma v-fold sur les sons. Illustration de l'algorithme heuristique pour la réduction du modèle par classe. Le seuil de précision du test par classe est fixé à $t_{\text{prcision}} = 80\%$ et la valeur initiale est $\mathbf{n}'^{(0)} = [400, 400, 400, 400, 400]$. Si la précision du test par classe est inférieure à t_{prcision}, n'_z diminue de 20, et si supérieure à n'_z augmente de 1.

	Itérations	Alarme $z=1$	Détonation $z=2$	Moteur $z=3$	Pas $z=4$	Voix $z=5$
n'_z	0	400	400	400	400	400
	1	401	401	380	401	380
	2	402	402	360	402	360
	3	403	382	340	403	340
Class-wise Accuracy (%)	0	98,6	84,2	66,9	80,9	54,3
	1	98,6	80,8	66,9	80,9	54,3
	2	98,6	77,3	68,1	80,9	54,3
	3	98,6	75,6	68,1	80,9	54,3

réduction jusqu'à 10, la précision augmente et pour les facteurs de réduction supérieurs à 50, elle diminue : cela peut signifier que la méthode a surappris quand il n'y a pas de réduction et que cette réduction tend à améliorer les résultats et donc diminuer le surapprentissage. Cependant, pour un facteur de réduction important, il y a moins de noyaux utilisés pour calculer la règle de classification et donc la précision diminue.

De plus le Tableau 3.10 donne une illustration de l'algorithme heuristique de réduction du modèle (seules les 4 premières itérations sont affichées). Ici, pour une réduction initiale importante, la classe Alarme est bien classée (plus de 98% de précision) contrairement à la classe Voix (54% de précision). Au fur et à mesure que la réduction est mise à jour, la précision de la classification augmente pour certaines classes (Moteur) ou diminue (Détonation).

Avantages et inconvénients. La méthode proposée présente plusieurs avantages par rapport aux méthodes de l'état de l'art. Un avantage est qu'il n'y a pas d'hyperparamètre à régler : la longueur de frame Δt et le facteur de quantification q n'influencent pas les résultats tant que q reste supérieur à B (qui peut être considéré comme «l'information statistique» disponible). La méthode proposée n'est pas la meilleure en terme de précision : un GMM bien réglé peut donner de meilleurs résultats. Cependant, le GMM doit avoir un nombre de composantes fixé à l'avance ou optimisé à l'aide d'un critère de sélection de modèle (comme le BIC ou le taux de bonne classification), ce qui augmente le temps d'apprentissage. Les DCNN ont une architecture de réseau complexe, difficile à régler et longue à entrainer. La méthode proposée présente ce second avantage que le temps d'apprentissage est réduit par rapport aux autres méthodes. Enfin, avec le prétraitement du modèle, la méthode proposée peut être utilisée en temps réel. En résumé, RARE est la deuxième meilleure méthode (juste après le GMM avec le bon réglage) et a la capacité de contrôler le compromis entre le temps de calcul et la précision *a posteriori* de l'apprentissage et de la réduction de complexité.

3.7.5 Résultats de la classification polyphonique

Méthode proposée sans réduction. Le Tableau 3.11 résume les résultats sur l'ensemble de données A-Volute pour la tâche de classification polyphonique de la même manière que pour la tâche de classification monophonique. Les données d'entraînement polyphoniques ont

été créées en considérant un ensemble de données d'entraînement monophonique réduit. En effet, le nombre de mélanges possibles à partir de l'ensemble de données d'origine était trop important pour pouvoir être stocké dans une mémoire d'ordinateur standard, de sorte que les mélanges sont créés en utilisant un ensemble de données réduit. Pour $r_{\text{mono}} = 400$, l'ensemble de données monophoniques est réduit d'un facteur 400 et des mélanges ont été créés en utilisant cet ensemble de données réduit. Ensuite, les mélanges ont été classés en utilisant les modèles monophoniques réduits et les modèles polyphoniques. Par conséquent, lorsque la réduction r_{mono} diminue, les scores deviennent meilleurs (puisque plus de modèles sont utilisés pour construire l'ensemble de données polyphoniques). Le CRNN doit être entrainé en utilisant un ensemble de données contenant des sons mixtes, c'est pourquoi des mélanges artificiels ont été créés sur l'ensemble de données A-Volute. Le paramètre N_{mixt} contrôle le nombre de frames de chaque classe mélangées ensemble : si $N_{\text{mixt}} = 200$, il y avait 200 frames de la première classe mélangées avec 200 frames de la seconde classe. De plus, une fois le réseau formé, un seuil doit être fixé : la valeur par défaut est 0,5 mais elle est totalement arbitraire. Ce tableau montre que la méthode RARE surpasse la méthode CRNN en termes de résultats et de temps de test. En effet, dans le cas $r_{\text{mono}} = 400$ et $r_{\text{poly}} = 1$, RARE obtient un F1 de 69,4% et un E.R. de 46,2% alors que le CRNN entrainé avec le plus de données ($N_{\text{mixt}} = 300$) obtient au plus un F1 de 53,1% et un E.R. de 60,3%.

Les Tableaux 3.12 et 3.13 résument les résultats pour le jeu de données Battlefield et TUT-SED comme précédemment. La principale différence entre ces ensembles de données et l'ensemble de données A-Volute est la présence de mélanges dans les ensembles de données. C'est pourquoi la méthode proposée n'utilise qu'un facteur de réduction pour les classes polyphoniques. Pour l'ensemble de données de Battlefield, la méthode proposée donne de meilleurs résultats en termes de scores (F1 et taux d'erreur) et de temps (entraînement et tests).

Méthode proposée avec réduction. Pour un r_{mono} plus petit, la méthode proposée pour l'ensemble de données A-Volute doit utiliser une étape de réduction polyphonique pour fonctionner en temps réel. Comme la réduction monophonique, la réduction polyphonique améliore les résultats de sorte que, sans réduction, la méthode proposée peut avoir surappris. L'ensemble de données de Battlefield montre un comportement similaire : de meilleurs scores pour une réduction d'un facteur 10, puis une diminution pour un facteur de réduction important. L'ensemble de données TUT-SED ne peut pas être réduit avec nos ordinateurs car le clustering hiérarchique utiliserait trop de RAM (de l'ordre de 120 Go).

Avantages et inconvénients. Toutes les méthodes précédentes utilisées pour la classification monophonique ne sont pas conçues pour travailler sur la classification polyphonique à partir d'un ensemble de données monophoniques, contrairement à la méthode que nous proposons, ce qui constitue un avantage majeur. En effet, la méthode proposée ne doit pas apprendre les mélanges de sons mais seulement les sons individuels.

3.8 Conclusion

Dans ce chapitre, nous avons présenté la méthode RARE qui permet de faire de la classification audio en temps-réel, monophonique et polyphonique. Cette méthode est basée sur un modèle génératif du spectre de puissance normalisé et une estimation de densité par noyau pour calculer la distribution conditionnelle nécessaire à la classification. Les contributions principales de cette méthode sont les suivantes :

Tableau 3.11 – Tâche de classification polyphonique. Résumé des résultats de l'ensemble de données A-Volute en termes de score F1 (F1), de taux d'erreur (E.R.), de temps d'entraînement (en secondes) et de temps de test par image (en millisecondes) pour différentes méthodes.

Méthode	Param.		F1	E.R.	Apprentissage	Test
RARE	$r_{\text{mono}} = 400$	$r_{\text{poly}} = 1$	69,4 (3,2)	46,2 (3,2)	$7,7 \times 10^1$	$6,4 \times 10^0$
	$r_{\text{mono}} = 100$	$r_{\text{poly}} = 1$	72,3 (4,8)	43,1 (5,4)	$1,9 \times 10^4$	$7,9 \times 10^2$
		$r_{\text{poly}} = 20$	71,4 (3,8)	41,3 (4,5)		$4,2 \times 10^1$
	$r_{\text{mono}} = 50$	$r_{\text{poly}} = 1$	74,5 (4,7)	40,6 (5,1)	$7,7 \times 10^1$	$3,0 \times 10^3$
CRNN [18]	$N_{\text{mixt}} = 200$	th = 0,5	39,8 (26,5)	70,3 (20,0)	$4,8 \times 10^3$	$2,8 \times 10^1$
		th = 0,2	56,4 (3,8)	114,4 (6,0)		
		th = 0,1	55,9 (2,1)	141,7 (15,0)		
CRNN [18]	$N_{\text{mixt}} = 300$	th = 0,5	53,1 (0,6)	60,3 (0,6)	$1,3 \times 10^4$	$3,1 \times 10^1$
		th = 0,2	56,2 (1,9)	141,1 (14,4)		
		th = 0,1	57,1 (0,1)	149,1 (1,9)		

Tableau 3.12 – Tâche de classification polyphonique. Résumé des résultats pour l'ensemble de données du Battlefield en termes de score F1 (F1), de taux d'erreur (E.R.), de temps d'entraînement (en secondes) et de temps de test par image (en millisecondes) pour différentes méthodes.

Méthode	Param.	F1	E.R.	Apprentissage	Test
RARE	$r_{\text{poly}} = 1$	66,0 (4,3)	44,5 (6,3)	$1,2 \times 10^2$	$9,4 \times 10^1$
	$r_{\text{poly}} = 10$	69,2 (2,8)	40,9 (2,6)		$1,0 \times 10^1$
	$r_{\text{poly}} = 50$	67,9 (1,8)	42,3 (2,2)		$2,1 \times 10^0$
CRNN [18]	th = 0,5	61,8 (3,4)	54,4 (4,4)	$1,8 \times 10^2$	$2,6 \times 10^1$
	th = 0,3	63,5 (4,0)	77,9 (17,3)		
	th = 0,2	65,2 (2,8)	85,1 (7,0)		
	th = 0,1	56,1 (1,5)	156,7 (9,4)		

Tableau 3.13 – Tâche de classification polyphonique. Résumé des résultats de l'ensemble de données TUT-SED en termes de score F1 (F1), de taux d'erreur (E.R.), de temps d'entraînement (en secondes) et de temps de test par image (en millisecondes) pour différentes méthodes.

Méthode	Param.	F1	E.R.	Apprentissage	Test
RARE	$r_{\text{poly}} = 1$	30,1 (4,1)	85,4 (9,8)	$5,4 \times 10^2$	$1,3 \times 10^3$
RARE	$r_{\text{poly}} = 100$	40,2 (9,8)	60,1 (9,7)	$8,8 \times 10^2$	$1,6 \times 10^1$
	$r_{\text{poly}} = 1000$	47,9 (11,9)	59,1 (11,3)	$8,8 \times 10^2$	$2,5 \times 10^0$
CRNN [18]	th = 0.,5	0,0 (0,0)	100,0 (0,0)	$2,3 \times 10^3$	$2,9 \times 10^1$
	th = 0,4	7,5 (15,0)	100,0 (0,0)		
	th = 0,3	25,2 (16,9)	99,1 (2,5)		
	th = 0,2	39,1 (7,7)	137,4 (57,9)		
	th = 0,1	34,5 (5,2)	308,4 (122,1)		

1. **Les données** : L'utilisation du spectre de puissance normalisé à la place des descripteurs audio usuels extraits des signaux audio. Ce spectre de puissance normalisé est particulièrement utile dans le cas polyphonique grâce à la propriété d'additivité des sources décorrélées.
2. **Modélisation** : Une modélisation générative très générale des spectres de puissance normalisés conçue pour le traitement temps-réel, qui utilise notamment la modélisation monophonique pour construire la modélisation polyphonique.
3. **Temps-réel** : Une réduction de complexité basée sur la classification hiérarchique des modèles de manière à obtenir un compromis précision - temps de calcul particulièrement efficace et pertinent.

Les expériences ont montré l'intérêt de notre méthode comparée aux autres techniques dans le cas de la classification temps-réel. De même la construction de l'ensemble d'apprentissage est très importante pour le temps-réel et influence beaucoup les résultats comparé aux méthodes classiques.

Du fait de l'estimation par noyaux notre méthode possède une flexibilité qu'il convient d'exploiter au mieux pour obtenir de bons résultats. Nous avons déjà considéré trois noyaux différents mais il semble qu'investiguer plus en détails sur le choix du noyau soit une piste intéressante de recherche pour cette méthode.

Les mathématiques sont l'art de donner le même nom à des choses différentes.

Henri Poincaré

Chapitre 4

Séparation de sources audio en temps-réel

Ce chapitre présente la méthode développée dans cette thèse pour la séparation de sources en temps-réel. Un état de l'art des modèles de sources et de mélanges ainsi que des méthodes de séparation est présentée dans les parties 4.1 et 4.2. La méthode développée dans cette thèse, appelée RASE pour *Real-time Audio Separation Engine*, est présentée en partie 4.3. Cette méthode utilise un modèle génératif ainsi que deux manières d'estimer les sources. La première proposition se place dans le contexte des données manquantes pour estimer les spectres des sources individuelles sachant le spectre observé et la distribution des sources. La seconde proposition est basée sur la déformation optimale de spectres de l'apprentissage. Les expériences numériques sont disponibles en partie 4.4. La partie 4.5 dresse la conclusion de ce chapitre.

4.1 Modèles de sources et de mélanges

4.1.1 Modèles de mélanges

Le mélange de sources sonores peut être modélisé de plusieurs manières. Un façon classique de décrire ce problème est de supposer que l'on dispose de J sources $\mathbf{s}(t) = \left(s_1(t), ..., s_J(t)^\top\right)$ qui se mélangent via M microphones distribués spatialement pour former le signal multi-canal $\mathbf{x}(t) = (x_1(t), ..., x_M(t)^\top)$. Le modèle de mélange de sources le plus simple est l'additivité dans le domaine temporel, dit *mélange instantané*.

Définition 18 (Mélange instantané)**.** *Le mélange instantané de plusieurs sources sonores est défini par :*

$$\mathbf{x}(t) = \mathbf{A}(t)\mathbf{s}(t) + \mathbf{n}(t), \tag{4.1}$$

avec $\mathbf{A}(t) \in \mathbb{R}^{M \times J}$ *la matrice de mélange qui contient les pondérations de chaque source et* $\mathbf{n}(t)$ *un bruit additif.*

La matrice $\mathbf{A}(t)$ est souvent considérée stationnaire, c'est-à-dire $\mathbf{A}(t) = \mathbf{A}$. Ce modèle est le plus simple possible, et historiquement fait référence au problème de *cocktail party* (plusieurs personnes discutent et leurs voix sont enregistrées par plusieurs microphones). Dans le cas où $J = M$, on peut estimer la matrice $\mathbf{A}$ avec la méthode ICA (Analyse en Composantes Indépendantes) [59].

Cependant ce modèle ne prend pas en compte d'éventuelles réverbérations ou interactions complexes entre les sources et les microphones. Ces phénomènes physiques peuvent être pris en compte par un modèle *convolutif*.

Définition 19 (Mélange convolutif)**.** *Le mélange de convolution de plusieurs sources sonores est défini par :*

$$\mathbf{x}(t) = \mathbf{A}(t) \star \mathbf{s}(t) + \mathbf{n}(t), \tag{4.2}$$

avec $\star$ l'opérateur de convolution.

Ces phénomènes physiques peuvent être exprimés par un modèle de source $x_j(t)$ prenant en compte la réponse impulsionnelle de la salle [95] :

$$x_j(t) = \sum_l h_{jk}(l) s_k(t-l), \tag{4.3}$$

avec $\mathbf{h}_{jk}$ les réponses impulsionnelles associées à la salle entre la source k et le microphone j.

Plus récemment, les modèles de mélanges considérés utilisent une représentation temps-fréquence (la STFT par exemple), de manière à changer les convolutions en produits :

$$\mathbf{x}(\tau, f) \approx \mathbf{A}(\tau, f)\mathbf{s}(\tau, f) + \mathbf{n}(\tau, f). \tag{4.4}$$

Finalement, la modélisation la plus souvent retenue est une somme non pondérée dans le domaine temps-fréquence d'*images spatiales de sources* $\mathbf{c}_j(\tau, f)$ [30] :

$$\mathbf{x}(\tau, f) = \sum_j \mathbf{c}_j(\tau, f). \tag{4.5}$$

Les modèles de sources présentés dans la partie suivantes sont basés sur ce dernier modèle (Eq. (4.5)).

4.1.2 Modèles de sources

Le modèle de source le plus courant suppose une distribution gaussienne complexe sur les bins temps-fréquence, dit *modèle gaussien local* [29].

Définition 20 (Modèle gaussien local)**.** *Le modèle gaussien local est défini par le modèle probabiliste suivant :*

$$\mathbf{c}_j(\tau, f) \sim \mathcal{N}_c\Big(\mathbf{0}, v_j(\tau, f)\mathbf{R}_j(f)\Big), \tag{4.6}$$

avec pour chaque source $v_j(\tau, f)\mathbf{R}_j(f)$ la matrice de covariance de la distribution, qui se décompose en deux termes : $v_j(\tau, f)$ la densité spectrale de puissance *(PSD) qui dépend de la frame et de la fréquence, et $\mathbf{R}_j(f)$ la matrice de* covariance spatiale *qui dépend de la fréquence (supposée la même au cours du temps).*

La matrice de covariance spatiale peut être décomposée de plusieurs manières. Elle peut notamment être décomposée pour exprimer un modèle convolutif, un modèle anéchoïque, un modèle de champ direct et diffus ainsi qu'un modèle non-contraint [28]. La densité spectrale de puissance peut être décomposée via un modèle filtre-excitation dont chacune des parties (filtre et excitation) est elle-même décomposée par une NMF, afin de modéliser les différents aspects temps-fréquence (bases spectrales et activations temporelles) [80].

Un autre type de modélisation probabiliste a été considéré, basé sur les *distributions α-stable* [101]. Ces distributions sont des distributions à queue lourde, qui généralisent des distributions classiques comme la distribution normale ou la distribution de Cauchy. Un cas particulier utilisé en séparation de sources est la distribution complexe isotropique α-stable, notée $\mathcal{S}\alpha\mathcal{S}_c(\cdot;\sigma)$ avec σ le paramètre de dispersion. Un autre cas a été considéré par Keriven [49] avec une distribution complexe symétrique centrée et de paramètre de dispersion unitaire notée $\mathcal{S}_c(\cdot;\alpha)$ avec α l'exposant caractéristique.

4.2 Etat de l'art des méthodes de séparation

La séparation de sources sonores est un domaine très étudié par la communauté du traitement du signal et de l'apprentissage statistique. Historiquement la première méthode utilisée était l'Analyse en composantes indépendantes (ICA) [59] qui cherchait à résoudre le problème cocktail party dans le cas déterminé (autant de sources que de micros). Depuis, beaucoup de techniques ont vu le jour, basées sur des méthodes d'optimisation ou d'apprentissage statistique supervisé.

Une première famille de méthode est la *factorisation en matrices / tenseurs non négatifs* (NMF / NTF). Comme vu précédemment, elle consiste à décomposer le spectrogramme (notamment $v_j(\tau, f)$) ou une autre représentation temps-fréquence (ou d'ordre supérieur) en un produit de matrices non négatives qui sont censées contenir les sources individuelles. Ozerov *et al.* [78] ont étudié le cas des mélanges convolutifs multicanal pour la séparation de sources musicales. Ils ont considéré un modèle gaussien complexe dont la PSD est décomposée par une NMF pour séparer différents instruments de musique et ont testé ce modèle à la fois dans les cas instantané et convolutif. Mirzaei *et al.* [73] ont également étudié le cas de mélanges sous-déterminés (c'est-à-dire moins de microphones que de sources) réverbérants. Ils ont proposé une NMF bayésienne pour factoriser les matrices de covariances des sources et une chaine de Markov inverse Gamma pour modéliser les dépendances temporelles dans le cas de la séparation de voix. Barker *et al.* [8] ont développé une méthode basée sur la décomposition des spectrogrammes de modulation (issus d'une banque de filtres) par des tenseurs non négatifs qui contiennent les contributions temporelles, fréquentielles, et par banque de filtre des sources, à la fois pour de la séparation musicale, de la réduction de bruit et de la séparation de voix. Mitsufuji *et al.* [74] ont appliqué la NTF dans le domaine du nombre d'onde en utilisant une méthode par noyaux pour estimer les covariances spatiales. Yoshii *et al.* [124] ont défini une nouvelle décomposition en tenseurs positifs semi-définis via la divergence de Bregman. Le choix du nombre de bases est déterminé avec une approche bayésienne, modélisé par un processus Gamma.

D'autres méthodes probabilistes ont été développées pour résoudre le problème de la séparation de sources. Une première méthode est appelée l'Analyse probabiliste en composantes latentes (PLCA). Duan *et al.* [26] ont proposé une version en ligne de la PLCA pour le temps-réel, et également une version semi-supervisée dans le cas où seulement une source est connue. Kim *et al.* [50] ont étendu la PLCA au cas «multicanal» quand plusieurs enregistrements sont disponibles (mais qui ne sont pas forcément issus de canaux spatiaux). Une deuxième méthode s'appelle la méthode FASST (*Flexible Audio Source Separation Toolbox*), présentée par Ozerov *et al.* dans [80]. Cette méthode se base sur une modélisation probabiliste de type gaussien avec matrice de covariance décomposée entre PSD et covariance spatiale, et spécifiquement la PSD est décomposée via deux NMF (une pour le modèle excitation-filtre, et l'autre pour le modèle localisé en temps / en bandes spectrales étroites). Simon *et al.* [100] ont proposé un estimateur en ligne pour FASST de manière à traiter en temps-réel la séparation de sources : cet estimateur consiste à considérer un «buffer» glissant de plusieurs frames qui servent à séparer les sources

puis à décaler ce buffer dans le temps. Le principal souci de cette estimation est que dans le cas non-supervisé, il faut résoudre le problème de la *permutation des sources* : suivant les frames les sources ne sont pas estimées «dans le même ordre».

Duong *et al.* [27] ont étudié plusieurs modèles de covariances et les estimateurs associés. Parmi les techniques étudiées, Vincent *et al.* [119] ont développé une modélisation parcimonieuse des coefficients temps-fréquence utilisant des priors gaussiens locaux dans le cas de mélanges instantanés stéréo. Sawada *et al.* [95] ont considéré un clustering par bin dans le cas d'un mélange complexe gaussien, et ont utilisé les probabilités *a posteriori* pour classer chaque bin temps-fréquence. Une étape de permutation permet d'améliorer les résultats de la séparation d'une frame à une autre. Kounades-Bastian *et al.* [52] ont proposé une modélisation par un modèle gaussien complexe dont la matrice de covariance est la PSD, dans un contexte bayésien (prior sur PSD, utilisation d'une inverse gamma) avec estimation par inférence variationnelle.

Plus récemment, l'apprentissage profond a été utilisé pour essayer de séparer les sources audio, soit en prédisant directement le spectre d'amplitude soit en estimant les densités spectrales de puissance. Nugraha *et al.* [76] se sont placés dans le cadre de FASST et ont utilisé un réseau de neurones dans l'algorithme EM pour estimer les PSD des sources. Zhang *et al.* [126] ont utilisé directement le signal audio temporel en entrée d'un CNN 1D qui prédit les frames des sources séparées. Grais *et al.* [36] ont développé une méthode en deux étapes : d'abord une séparation puis une amélioration des sources estimées, via des DNN et/ou des NMF dans chaque étape. Zhang *et al.* [127] ont considéré la séparation de voix en binaural avec un fond diffus spatialement distribué. La méthode fait un traitement spatial avec un beamforming pour focaliser sur une zone où la voix est localisée puis un DNN estime le spectre de la voix.

Certaines méthodes ont été étudiées pour fonctionner en temps-réel. Luo *et al.* [62] ont utilisé un autoencodeur sur le mélange temporel : l'encodeur est un CNN, la séparation se fait avec un RNN avec des cellules LSTM et le décodeur utilise les masques estimés par le séparateur et les vecteurs de bases pour reconstruire les sources. Naithani *et al.* [75] ont développé un réseau à faible latence qui utilise quelques frames de contexte en entrée d'un DNN qui prédit les masques des sources. Une variante des réseaux de neurones appelée GAN (*Generative Adversarial Network*) a également été considérée pour la séparation de sources grâce à sa propriété de générer de nouveaux échantillons à partir de la distribution générative apprise. Stoller *et al.* [106] ont considéré un GAN semi-supervisé qui à partir du spectrogramme du mélange essaie de séparer la voix des accompagnements et un discriminateur tente de discerner si les sources estimées sont correctes. Zhang *et al.* [125] ont proposé une séparation non supervisée avec GAN basée sur la distance de Wasserstein et une hypothèse de conservation de l'énergie du spectre. Uhlich *et al.* ont proposé des réseaux à 4 couches pour séparer les instruments de musique, pour les signaux mono (des DNN classiques) [115] et stéréo (avec des LSTM) [114]. Une étape d'agrégation des résultats de chaque réseau permet d'améliorer les réseaux.

Basé sur ces deux réseaux, Takahashi *et al.* ont utilisé des *DenseNet* seuls [108] et en agrégation avec un réseau LSTM [109]. Un bloc dense (*DenseBlock*, illustré en Figure 4.1(a)) est composé de plusieurs couches «composantes» (souvent au nombre de 4), et chaque couche composante est formée d'un Batch Normalisation, un ReLU et une convolution. L'intérêt de ces blocs denses est la présence de connexions de saut (*skip connections*) des couches précédentes sur les couches suivantes qui permet de supprimer les singularités qui peuvent apparaître pendant l'apprentissage [77]. L'implémentation de Takahashi utilise des blocs denses au sein d'un DenseNet particulier, appelé *MDenseNet* (illustré en Figure 4.1(b)) : entre chaque bloc dense, une étape de sous-échantillonnage (au sens signal) permet de réduire la dimension de la couche suivante. Après quelques couches et étapes de sous-échantillonnage, des étapes de sur-échantillonnage sont ajoutées entre chaque bloc dense, de manière à ce que la sortie du MDenseNet ait la même

dimension que l'entrée du réseau. Des connexions de saut sont ajoutées de manière à garder une certaine cohérence avec les blocs denses. Finalement, l'idée de Takahashi est d'utiliser un MDenseNet par bande de fréquence dans le spectrogramme de manière à le spécialiser dans chaque bande de fréquence (illustré en Figure 4.1(c)), d'où le nom *MMDenseNet* pour *Multi-scale Multi-band DenseNet*.

Parmi toutes les méthodes présentées plus haut, les méthodes à base de réseaux de neurones sont celles qui semblent présenter les meilleurs espoirs en terme de performance. Le principal problème de ces méthodes est le fait d'avoir besoin de bases de données conséquentes pour bien estimer les paramètres du réseau de neurones. À l'inverse, les méthodes probabilistes peuvent s'affranchir de l'utilisation de grosses bases de données pour fonctionner : c'est le cas de notre méthode de classification présentée au Chapitre 3, et c'est le cas de notre méthode de séparation également.

Méthodes oracles [1]. Étant donné le type de problème à résoudre, la communauté de la séparation de sources a défini plusieurs méthodes dites *oracles*. Une méthode est dite oracle lorsqu'elle prend en entrée la vérité terrain. Cela permet d'avoir une borne supérieure de comparaison avec les autres méthodes de séparation [107] utilisant le même principe (par exemple un type de masquage temps-fréquence). Par exemple, les méthodes oracles définies par Stöter [107] sont :

— MIX : la source estimée est le mélange divisé par le nombre de sources :

$$\widehat{\mathbf{s}}_j(\tau, f) = \frac{\mathbf{x}(\tau, f)}{J}. \tag{4.7}$$

— IBM (*Ideal Binary Mask*) 1 et 2 : le masque idéal binaire où le spectre d'amplitude ($\alpha = 1$) ou de puissance ($\alpha = 2$) est utilisé :

$$\mathbf{M}_j(\tau, f) = 1 \text{ si } \frac{\left|\mathbf{s}_j(\tau, f)\right|^\alpha}{\sum_{j'} \left|\mathbf{s}_{j'}(\tau, f)\right|^\alpha} > 0.5, \tag{4.8}$$

$$\widehat{\mathbf{s}}_j(\tau, f) = \mathbf{M}_j(\tau, f) \cdot \mathbf{x}(\tau, f). \tag{4.9}$$

— IRM (*Ideal Ratio Mask*) 1 et 2 : le masque idéal de ratio où le spectre d'amplitude ($\alpha = 1$) ou de puissance ($\alpha = 2$) est utilisé :

$$\mathbf{M}_j(\tau, f) = \frac{\left|\mathbf{s}_j(\tau, f)\right|^\alpha}{\sum_{j'} \left|\mathbf{s}_{j'}(\tau, f)\right|^\alpha}, \tag{4.10}$$

$$\widehat{\mathbf{s}}_j(\tau, f) = \mathbf{M}_j(\tau, f) \cdot \mathbf{x}(\tau, f). \tag{4.11}$$

4.3 Séparation de sources en temps-réel.

4.3.1 Énoncé du problème

Nous allons présenter la méthode de séparation audio en temps-réel développée dans cette thèse, appelée RASE (*Real-time Audio Separation Engine*). L'objectif est d'estimer au temps t les

1. Les méthodes oracles en séparation de sources sont légèrement différentes de celles en statistique. Une des différences majeures est le fait que cette méthode oracle ne produit pas forcément des résultats dont la métrique est une borne supérieure pour les autres méthodes, contrairement à ce qui se fait en statistique.

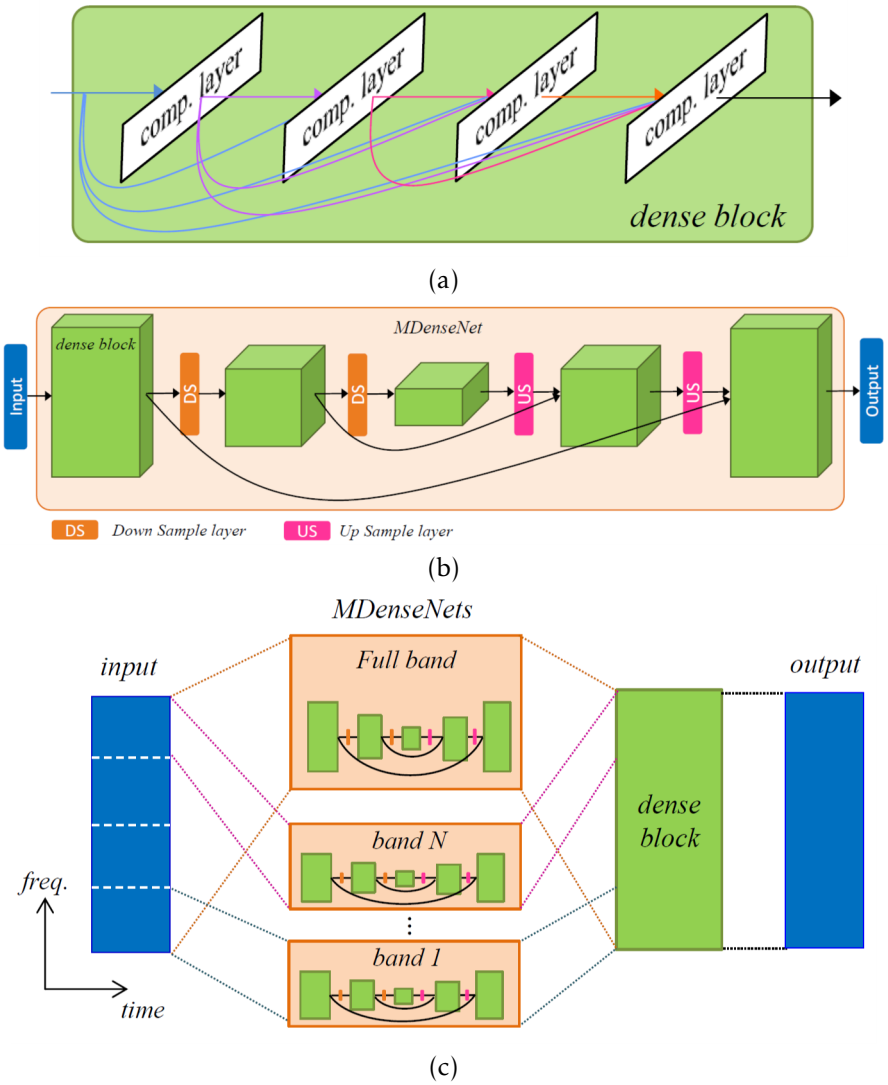


FIGURE 4.1 – Les élements constituant un MMDenseNet [108]). (a) Un bloc dense, composante de base d'un DenseNet. Celui-ci comporte 4 couches «composantes» (*comp. layer*) formées chacune d'un Batch Normalisation, un ReLU et une convolution. De plus, des connexions de sauts sont ajoutées venant des couches «composantes» précédentes (flèches bleues, mauves et rouges) sur les suivantes. (b) La structure du MDenseNet (Multi-scale DenseNet). Ce réseau est composé de 5 blocs denses (cubes verts) et comporte des couches de sous-échantillonnage (DS, carré orange) et de sur-échantillonnage (US, carré magenta) qui permettent de réduire puis augmenter la dimension des éléments dans le réseau. Des connexions de sauts sont également présentes comme dans les blocs denses. (c) La structure du MMDenseNet (Multi-band MDenseNet). Le spectrogramme est divisé en plusieurs bandes fréquentielles et chaque bande du spectrogramme est passée dans un MDenseNet particulier. Les sorties de tous les MDenseNets sont ensuite concaténées avant de passer dans un bloc dense pour ensuite prédire les sources séparées.

sources présentes dans un mélange instantané via une méthode probabiliste. On considère deux signaux notés $\mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]}$ et $\mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}$ observés sur une durée ΔT dont seule la somme est connue :

$$\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]} = \mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]} + \mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}. \tag{4.12}$$

Le signal $\mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]}$ est supposé provenir d'une classe de sons dont le label est z_1 (coup de feu, voix par exemple), et $\mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}$ est supposé provenir de la classe de label z_2. Ce mélange est considéré comme instantané, et on suppose que les sources peuvent contenir l'information de réverbération.

Définition 21 (Séparation de source)**.** *L'objectif de séparer à un instant t deux sources d'un mélange peut être écrit de la manière suivante :*

$$\widehat{\mathbf{x}}^{temp}_{1[t-\Delta T,t]} = f^{(-1)} \circ \mathcal{R} \circ f\left(\mathbf{x}^{temp}_{[t-\Delta T,t]}\right), \tag{4.13}$$

$$\widehat{\mathbf{x}}^{temp}_{2[t-\Delta T,t]} = \mathbf{x}^{temp}_{[t-\Delta T,t]} - \widehat{\mathbf{x}}^{temp}_{1[t-\Delta T,t]}, \tag{4.14}$$

avec :

- $\widehat{\mathbf{x}}^{\text{temp}}_{1[t-\Delta T,t]}$ $\left(\text{respectivement } \widehat{\mathbf{x}}^{\text{temp}}_{2[t-\Delta T,t]}\right)$ l'estimation de la source $\mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]}$ $\left(\text{respectivement } \mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}\right)$,
- $f\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right)$ la fonction qui calcule les attributs de $\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}$ utiles pour la séparation,
- $\mathcal{R}(\cdot)$ la fonction qui estime la source de la classe 1 séparée à partir du mélange,
- et $f^{(-1)}(\cdot)$ la fonction réciproque de $f(\cdot)$ qui correspond à la reconstruction des signaux.

De même que pour la classification de sources, les signaux sont observés sur un temps d'observation ΔT et sont acquis à chaque multiple de δt. L'objectif est donc de trouver des attributs intéressants pour la séparation ainsi que la fonction de séparation, qui seront différents de la classification. La principale différence par rapport à la classification est également le fait que l'on cherche à prédire un signal et non un label, et donc l'information à retrouver est beaucoup plus complexe.

4.3.2 Conception des attributs

Les attributs pertinents pour la séparation sont légèrement différents que ceux de la classification. En effet, la séparation de sources audio consiste à prédire des signaux sonores, dont le volume et la phase sont des éléments importants. Ainsi, le spectre de puissance normalisé ne sera pas considéré ici. L'attribut utilisé sera directement le spectre complexe issu d'un découpage en frames des sons. En effet, de part le contexte temps-réel, on considère des signaux transformés en frames :

$$\left(\mathbf{x}^{\text{temp}}_{1t},\ldots,\mathbf{x}^{\text{temp}}_{Nt}\right) = f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{temp}}_{[t-\Delta T,t]}\right), \tag{4.15}$$

$$= f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{temp}}_{1[t-\Delta T,t]}\right) + f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{temp}}_{2[t-\Delta T,t]}\right), \tag{4.16}$$

avec $f^{(\Delta t)}_{\text{frame}}(\cdot)$ opération de découpe d'un son en frames de taille Δt décalé d'un multiple de δt, et + l'addition élément par élément. Chaque frame est un bout de son de taille Δt décalé d'un multiple de δt et fenêtré avec une fenêtre $\mathbf{w}$:

$$\mathbf{x}_{\tau t}^{\text{temp}} = \mathbf{w} \cdot \mathbf{x}_{[t-\Delta T+\tau\delta t, t-\Delta T+\tau\delta t+\Delta t]}^{\text{temp}},\ \tau = 1, ..., N. \tag{4.17}$$

De plus, on se place dans le domaine fréquentiel avec la Transformée de Fourier (TF). Ainsi la fonction qui calcule les attributs est la composée du découpage en frame et de la TF :

$$\begin{aligned}
\left(\mathbf{x}_{1t}^{\text{freq}}, ..., \mathbf{x}_{Nt}^{\text{freq}}\right) &= f\left(\mathbf{x}_{[t-\Delta T,t]}^{\text{temp}}\right) && (4.18)\\
&= f_{\text{TF}} \circ f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{[t-\Delta T,t]}^{\text{temp}}\right), && (4.19)\\
&= f_{\text{TF}} \circ f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{1[t-\Delta T,t]}^{\text{temp}}\right) + f_{\text{TF}} \circ f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{2[t-\Delta T,t]}^{\text{temp}}\right), && (4.20)
\end{aligned}$$

avec $f_{\text{TF}}(\cdot)$ l'opérateur de la transformée de Fourier sur chaque frame $\mathbf{x}_{\tau t}^{\text{temp}}$. La séparation se fera directement sur le spectre complexe car contrairement à la classification, l'information de volume ainsi que de phase est importante dans ce contexte.

Remarque : fonction d'attribut réciproque. La fonction de reconstruction présente dans la définition de la séparation est la fonction d'attribut réciproque :

$$f^{(-1)}\left(\mathbf{x}_{1t}^{\text{freq}}, ..., \mathbf{x}_{Nt}^{\text{freq}}\right) = \left(f_{\text{frame}}^{(\Delta t)}\right)^{(-1)} \circ f_{\text{TF}}^{(-1)}\left(\mathbf{x}_{1t}^{\text{freq}}, ..., \mathbf{x}_{Nt}^{\text{freq}}\right), \tag{4.21}$$

avec $f_{\text{TF}}^{(-1)}$ la transformée de Fourier inverse et $\left(f_{\text{frame}}^{(\Delta t)}\right)^{(-1)}$ l'overlap-add (présenté en Chapitre 2).

4.3.3 Conception du modèle

Le processus génératif pour la séparation est sensiblement identique au processus génératif pour la classification polyphonique, hormis une étape supplémentaire et le calcul des attributs. Le processus génératif des données pour la séparation est le suivant :

- **Aléatoire** : Tirer sans remise les classes $z_1, z_2 \sim \text{Mult}_K(\cdot; 1, \mathbf{p})$,
- **Aléatoire** : Tirer sans remise les indices des sons à tirer $y_1 \sim p\left(\cdot \middle| z_1\right)$ et $y_2 \sim p\left(\cdot \middle| z_2\right)$,
- **Aléatoire** : Tirer les sons $\mathbf{x}_1^{\text{temp}} \sim p_{\text{sound}}\left(\cdot \middle| y_1, z_1\right)$ et $\mathbf{x}_2^{\text{temp}} \sim p_{\text{sound}}\left(\cdot \middle| y_2, z_2\right)$,
- **Aléatoire** : Tirer un temps $t \sim \mathcal{U}([0, T])$,
- **Déterministe** : Mélange des spectres complexes $\left(\mathbf{x}_{1t}^{\text{freq}}, ..., \mathbf{x}_{Nt}^{\text{freq}}\right) = f_{\text{TF}} \circ f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{1[t-\Delta T,t]}^{\text{temp}}\right) + f_{\text{TF}} \circ f_{\text{frame}}^{(\Delta t)}\left(\mathbf{x}_{2[t-\Delta T,t]}^{\text{temp}}\right)$.

La variable z correspond à un label de classe de sons considérés (voix, coup de feu,...), alors que la variable y correspond à un «label» de son dans la base d'apprentissage. La nouvelle idée ici est de considérer l'indice des sons desquels sont issus les signaux mélangés. En effet, ceux-ci seront utiles dans le cas des déformations optimales (proposition 2 de la sous-partie 4.3.5). Le processus génératif modélise la distribution jointe $p\left(\mathbf{x}_{1t}^{\text{freq}}, ..., \mathbf{x}_{Nt}^{\text{freq}}, t, \mathbf{x}_1^{\text{temp}}, \mathbf{x}_2^{\text{temp}}, y_1, y_2, z_1, z_2\right)$.

Dans la suite du chapitre, deux manières de résoudre le problème de séparation seront présentées : par données manquantes (proposition 1, sous-partie 4.3.4) et par déformation optimale (proposition 2, sous-partie 4.3.5). Chacune des propositions utilisera de manière différente la probabilité jointe précédente et donnera donc des fonctions de séparation $\mathcal{R}$ différentes. L'idée de base des deux propositions est de considérer la distribution conditionnelle des spectres sachant le mélange et les classes $p\left(\mathbf{x}_{11t}^{\text{freq}}, ..., \mathbf{x}_{1Nt}^{\text{freq}}, \mathbf{x}_{21t}^{\text{freq}}, ..., \mathbf{x}_{2Nt}^{\text{freq}} \middle| \mathbf{x}_{1t}^{\text{freq}}, ..., \mathbf{x}_{Nt}^{\text{freq}}, t, y_1, y_2, z_1, z_2\right)$.

Plusieurs hypothèses sont faites sur cette distribution, similaires à celles de la classification. Dans un premier temps, une hypothèse d'indépendance conditionnelle entre les frames :

$$p\left(\mathbf{x}_{11t}^{\text{freq}},...,\mathbf{x}_{1Nt}^{\text{freq}},\mathbf{x}_{21t}^{\text{freq}},...,\mathbf{x}_{2Nt}^{\text{freq}}\middle|\mathbf{x}_{1t}^{\text{freq}},...,\mathbf{x}_{Nt}^{\text{freq}},t,z_1,z_2\right)=\prod_{\tau=1}^{N}p\left(\mathbf{x}_{1\tau t}^{\text{freq}},\mathbf{x}_{2\tau t}^{\text{freq}}\middle|\mathbf{x}_{\tau t}^{\text{freq}},t,z_1,z_2\right). \tag{4.22}$$

Cette hypothèse permettra de simplifier le traitement effectué dans les parties suivantes. De plus, l'hypothèse de stationnarité en temps permet de retirer l'influence de l'indice de temps t sur la distribution :

$$p\left(\mathbf{x}_{1\tau t}^{\text{freq}},\mathbf{x}_{2\tau t}^{\text{freq}}\middle|\mathbf{x}_{\tau t}^{\text{freq}},t,z_1,z_2\right)=p\left(\mathbf{x}_{1\tau}^{\text{freq}},\mathbf{x}_{2\tau}^{\text{freq}}\middle|\mathbf{x}_{\tau}^{\text{freq}},z_1,z_2\right). \tag{4.23}$$

Finalement, dans un premier temps nous travaillerons à classes fixées, de sorte que :

$$p\left(\mathbf{x}_{1\tau}^{\text{freq}},\mathbf{x}_{2\tau}^{\text{freq}}\middle|\mathbf{x}_{\tau}^{\text{freq}},z_1,z_2\right)=p\left(\mathbf{x}_{1\tau}^{\text{freq}},\mathbf{x}_{2\tau}^{\text{freq}}\middle|\mathbf{x}_{\tau}^{\text{freq}}\right). \tag{4.24}$$

Cette dernière équation est surtout une simplification d'écriture (et non une indépendance) car en pratique les signaux à séparer appartiennent bien à une classe particulière.

4.3.4 Proposition 1 : Estimation par données manquantes (DM-GMM)

Formalisation. Cette proposition modélise de manière probabiliste la distribution de chaque source ainsi que la distribution conditionnelle d'une source (la première par exemple) sous l'hypothèse de somme des deux sources. Cette formulation a été proposé de manière similaire dans les articles [79] et [9]. Ceux-ci considèrent un cadre d'estimation bayésien où les moyennes *a priori* des sources sont supposées nulles et un bruit faible est ajouté au mélange. Néanmoins la proposition de cette thèse est présentée car elle utilise un cadre assez similaire à la méthode de classification et permet de faire le lien avec la deuxième proposition présentée plus loin. Ici, l'hypothèse de stationnarité entre les frames est faite sur la distribution conditionnelle :

$$p\left(\mathbf{x}_{1\tau}^{\text{freq}},\mathbf{x}_{2\tau}^{\text{freq}}\middle|\mathbf{x}_{\tau}^{\text{freq}}\right)=p\left(\mathbf{x}_{1\tau'}^{\text{freq}},\mathbf{x}_{2\tau'}^{\text{freq}}\middle|\mathbf{x}_{\tau'}^{\text{freq}}\right). \tag{4.25}$$

Toutes les hypothèses précédentes permettent d'écrire la distribution conditionnelle de la manière suivante :

$$p\left(\mathbf{x}_{1\tau}^{\text{freq}},\mathbf{x}_{2\tau}^{\text{freq}}\middle|\mathbf{x}_{\tau}^{\text{freq}}\right)=p\left(\mathbf{x}_{1}^{\text{freq}},\mathbf{x}_{2}^{\text{freq}}\middle|\mathbf{x}^{\text{freq}}\right). \tag{4.26}$$

Les attributs étant le spectre complexe, ces derniers sont convertis en vecteurs de partie réelle et imaginaire sous la forme $\overline{\mathbf{x}}^{\text{freq}}=[\text{Re}(\mathbf{x}^{\text{freq}});\text{Im}(\mathbf{x}^{\text{freq}})]$. L'additivité des spectres est conservée après cette transformation de sorte que :

$$\overline{\mathbf{x}}^{\text{freq}}=\overline{\mathbf{x}}_1^{\text{freq}}+\overline{\mathbf{x}}_2^{\text{freq}}. \tag{4.27}$$

Modèle de sources. Le modèle de base pour cette proposition est une distribution de la forme modèle de mélanges sur $\overline{\mathbf{x}}_1^{\text{freq}}$ et $\overline{\mathbf{x}}_2^{\text{freq}}$:

$$p_z\left(\overline{\mathbf{x}}_z^{\text{freq}}\right)=\sum_{g_z=1}^{G_z}\pi_{zg_z}p_z\left(\overline{\mathbf{x}}_z^{\text{freq}};\boldsymbol{\theta}_{g_z}\right). \tag{4.28}$$

avec $\left(\boldsymbol{\theta}_{g_z}\right)_{g_z=1}^{G_z}$ les paramètres des composantes du mélange et $\left(\pi_{zg_z}\right)_{g_z=1}^{G_z}$ les proportions du

mélange telles que $\pi_{zg_z} > 0$ et $\sum_{g_z=1}^{G_z} \pi_{zg_z} = 1$. En effet l'utilisation de loi mélange apporte une certaine flexibilité dans la modélisation des données. De plus, nous verrons dans la suite que la forme finale des distributions considérées sont également de la forme loi mélange, ce qui en fait un outil très simple à utiliser dans un premier temps. Cette loi (4.28) est estimée, pour chaque classe z, sur les spectres de l'ensemble d'apprentissage $\overline{\mathbf{x}}_{z(i)}^{\text{freq}}$. Le nombre de composantes G_z est soit fixé à l'avance, soit déterminé par un critère de sélection de modèles (BIC par exemple).

Problème à données manquantes. L'idée est de considérer la distribution conditionnelle de $\overline{\mathbf{x}}_1^{\text{freq}}$ sachant $\overline{\mathbf{x}}^{\text{freq}}$, sous hypothèse que $\overline{\mathbf{x}}_1^{\text{freq}}$ et $\overline{\mathbf{x}}_2^{\text{freq}}$ sont indépendants (de part le modèle génératif) :

$$p\left(\overline{\mathbf{x}}_1^{\text{freq}} \middle| \overline{\mathbf{x}}^{\text{freq}}\right) = \frac{p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}\right) p_2\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_1^{\text{freq}}\right)}{p\left(\overline{\mathbf{x}}^{\text{freq}}\right)}. \tag{4.29}$$

Dans le cas de la loi mélange décrite plus haut, le produit des distributions et la distribution de $\overline{\mathbf{x}}^{\text{freq}}$ sont calculés de la manière suivante :

$$p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}\right) p_2\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_1^{\text{freq}}\right) = \sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \pi_{1g_1} \pi_{2g_2} p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_1}\right) p_2\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_2}\right), \tag{4.30}$$

$$p\left(\overline{\mathbf{x}}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \pi_{1g_1} \pi_{2g_2} \int_{\overline{\mathbf{x}}_1^{\text{freq}}} p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_1}\right) p_2\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_2}\right) \mathrm{d}\overline{\mathbf{x}}_1^{\text{freq}}. \tag{4.31}$$

Ainsi la distribution conditionnelle s'écrit :

$$p\left(\overline{\mathbf{x}}_1^{\text{freq}} \middle| \overline{\mathbf{x}}^{\text{freq}}\right) = \frac{\sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \pi_{1g_1} \pi_{2g_2} p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_1}\right) p_2\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_2}\right)}{\sum_{g_1'=1}^{G_1} \sum_{g_2'=1}^{G_2} \pi_{1g_1'} \pi_{2g_2'} \int_{\overline{\mathbf{x}}_1^{\text{freq}}} p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_1'}\right) p_2\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_1^{\text{freq}}; \boldsymbol{\theta}_{g_2'}\right) \mathrm{d}\overline{\mathbf{x}}_1^{\text{freq}}}. \tag{4.32}$$

Utilisation des GMM. Les lois mélanges sur les spectres sont supposées être des GMM, ce qui implique que chaque composante est une gaussienne :

$$p\left(\overline{\mathbf{x}}_z^{\text{freq}}; \boldsymbol{\theta}_{g_z}\right) = \mathcal{N}\left(\overline{\mathbf{x}}_z^{\text{freq}}; \boldsymbol{\mu}_{zg_z}, \Sigma_{zg_z}\right). \tag{4.33}$$

En effet ces distributions ont de bonnes propriétés statistiques et permettent en première approche de réaliser des calculs simplifiés par rapport à d'autres distributions. La propriété d'additivité sous indépendance des variables aléatoires normales permet d'écrire la distribution du mélange $\overline{\mathbf{x}}^{\text{freq}}$ de la manière suivante :

$$p\left(\overline{\mathbf{x}}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \pi_{1g_1} \pi_{2g_2} \mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}; \boldsymbol{\mu}_{1g_1} + \boldsymbol{\mu}_{2g_2}, \Sigma_{1g_1} + \Sigma_{2g_2}\right). \tag{4.34}$$

La distribution conditionnelle s'écrit quant à elle sous la forme suivante :

$$p\left(\overline{\mathbf{x}}_1^{\text{freq}}\middle|\overline{\mathbf{x}}^{\text{freq}}\right) = \frac{\sum_{g_1=1}^{G_1}\sum_{g_2=1}^{G_2}\pi_{1g_1}\pi_{2g_2}\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1}+\boldsymbol{\mu}_{2g_2},\Sigma_{1g_1}+\Sigma_{2g_2}\right)\frac{\mathcal{N}\left(\overline{\mathbf{x}}_1^{\text{freq}};\boldsymbol{\mu}_{1g_1},\Sigma_{1g1}\right)\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}-\overline{\mathbf{x}}_1^{\text{freq}};\boldsymbol{\mu}_{2g_2},\Sigma_{2g_2}\right)}{\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1}+\boldsymbol{\mu}_{2g_2},\Sigma_{1g_1}+\Sigma_{2g_2}\right)}}{\sum_{g_1'=1}^{G_1}\sum_{g_2'=1}^{G_2}\pi_{1g_1'}\pi_{2g_2'}\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1'}+\boldsymbol{\mu}_{2g_2'},\Sigma_{1g_1'}+\Sigma_{2g_2'}\right)\int_{\mathbf{x}_1}\frac{\mathcal{N}\left(\mathbf{x}_1;\boldsymbol{\mu}_{1g_1'},\Sigma_{1g_1'}\right)\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}-\mathbf{x}_1;\boldsymbol{\mu}_{2g_2'},\Sigma_{2g_2'}\right)}{\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1'}+\boldsymbol{\mu}_{2g_2'},\Sigma_{1g_1'}+\Sigma_{2g_2'}\right)}\mathrm{d}\mathbf{x}_1}. \tag{4.35}$$

Cette distribution conditionnelle fait apparaitre la distribution conditionnelle de deux gaussiennes sous la forme d'une fraction, ce qui donne le résultat classique :

$$\frac{\mathcal{N}\left(\overline{\mathbf{x}}_1^{\text{freq}};\boldsymbol{\mu}_{1g_1},\Sigma_{1g_1}\right)\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}-\overline{\mathbf{x}}_1^{\text{freq}};\boldsymbol{\mu}_{2g_2},\Sigma_{2g_2}\right)}{\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1}+\boldsymbol{\mu}_{2g_2},\Sigma_{1g_1}+\Sigma_{2g_2}\right)} = \mathcal{N}\left(\overline{\mathbf{x}}_1^{\text{freq}};\tilde{\boldsymbol{\mu}}_{g_1g_2},\tilde{\Sigma}_{g_1g_2}\right), \tag{4.36}$$

avec :

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_{g_1g_2} &= \boldsymbol{\mu}_{1g_1}+\Sigma_{1g_1}\left(\Sigma_{1g_1}+\Sigma_{2g_2}\right)^{-1}\left(\overline{\mathbf{x}}^{\text{freq}}-\boldsymbol{\mu}_{1g_1}-\boldsymbol{\mu}_{2g_2}\right), &(4.37)\\ \tilde{\Sigma}_{g_1g_2} &= \Sigma_{1g_1}-\Sigma_{1g_1}\left(\Sigma_{1g_1}+\Sigma_{2g_2}\right)^{-1}\Sigma_{1g_1}. &(4.38)\end{aligned}$$

De plus le dénominateur se simplifie avec l'intégrale de la densité conditionnelle de deux gaussiennes (qui vaut 1). On note :

$$\varphi_{g_1g_2}\left(\overline{\mathbf{x}}^{\text{freq}}\right) = \frac{\pi_{1g_1}\pi_{2g_2}\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1}+\boldsymbol{\mu}_{2g_2},\Sigma_{1g_1}+\Sigma_{2g_2}\right)}{\sum_{g_1'=1}^{G_1}\sum_{g_2'=1}^{G_2}\pi_{1g_1'}\pi_{2g_2'}\mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}};\boldsymbol{\mu}_{1g_1'}+\boldsymbol{\mu}_{2g_2'},\Sigma_{1g_1'}+\Sigma_{2g_2'}\right)}. \tag{4.39}$$

Ainsi la distribution conditionnelle recherchée peut se réécrire comme une distribution mélange :

$$p\left(\overline{\mathbf{x}}_1^{\text{freq}}\middle|\overline{\mathbf{x}}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1}\sum_{g_2=1}^{G_2}\varphi_{g_1g_2}\left(\overline{\mathbf{x}}^{\text{freq}}\right)\mathcal{N}\left(\overline{\mathbf{x}}_1^{\text{freq}};\tilde{\boldsymbol{\mu}}_{g_1g_2},\tilde{\Sigma}_{g_1g_2}\right). \tag{4.40}$$

Reconstruction des sources. Une estimation du spectre vectorisé de la frame τ de la source 1 peut être calculée avec l'espérance conditionnelle de la distribution précédente :

$$\begin{aligned}\widehat{\overline{\mathbf{x}}}_{1\tau}^{\text{freq}} &= \mathbb{E}_{p\left(\cdot\middle|\overline{\mathbf{x}}_\tau^{\text{freq}}\right)}\left[\mathbf{X}_{1\tau}\middle|\mathbf{X}=\overline{\mathbf{x}}_\tau^{\text{freq}}\right], &(4.41)\\ &= \sum_{g_1=1}^{G_1}\sum_{g_2=1}^{G_2}\varphi_{g_1g_2}\left(\overline{\mathbf{x}}_\tau^{\text{freq}}\right)\left(\boldsymbol{\mu}_{1g_1}+\Sigma_{1g_1}\left(\Sigma_{1g_1}+\Sigma_{2g_2}\right)^{-1}\left(\overline{\mathbf{x}}_\tau^{\text{freq}}-\boldsymbol{\mu}_{1g_1}-\boldsymbol{\mu}_{2g_2}\right)\right). &(4.42)\end{aligned}$$

Une estimation du spectre complexe de la source 1 se trouve en recombinant partie réelle et partie imaginaire :

$$\widehat{x}_{1\tau b}^{\text{freq}} = \widehat{\overline{x}}_{1\tau,b}^{\text{freq}} + i\widehat{\overline{x}}_{1\tau,b+B}^{\text{freq}},\ b=1,\ldots,B. \tag{4.43}$$

La fonction de séparation consiste donc en les espérances conditionnelles prises en chaque spectre (de chaque frame) :

$$\widehat{\mathcal{R}}\left(\overline{\mathbf{x}}_1^{\text{freq}},...,\overline{\mathbf{x}}_N^{\text{freq}}\right) = \left(\mathbb{E}_{p\left(\cdot|\overline{\mathbf{x}}_1^{\text{freq}}\right)}\left[\mathbf{X}_{11}\middle|\mathbf{X}=\overline{\mathbf{x}}_1^{\text{freq}}\right],...,\mathbb{E}_{p\left(\cdot|\overline{\mathbf{x}}_N^{\text{freq}}\right)}\left[\mathbf{X}_{1N}\middle|\mathbf{X}=\overline{\mathbf{x}}_N^{\text{freq}}\right]\right). \tag{4.44}$$

La séparation réalisée avec la proposition 1 peut donc être vue comme une régression issue d'un modèle génératif à base de modèle de mélanges gaussien (appelé *Gaussian mixture regression*) :

$$\overline{\mathbf{x}}_1^{\text{freq}} = \mathbf{R}_1\overline{\mathbf{x}}^{\text{freq}} + \boldsymbol{\epsilon}_1, \tag{4.45}$$

avec $\boldsymbol{\epsilon}_1$ un bruit additif et $\mathbf{R}_1$ le modèle de régression linéaire.

Remarque : Extension dans le cas à plus de deux sources. Dans le cas d'un mélange de trois sources, la méthode consiste à considérer un calcul *hiérarchique* des distributions. On considère le mélange suivant :

$$\begin{aligned}\overline{\mathbf{x}}^{\text{freq}} &= \overline{\mathbf{x}}_1^{\text{freq}} + \overline{\mathbf{x}}_2^{\text{freq}} + \overline{\mathbf{x}}_3^{\text{freq}}, &(4.46)\\ &= \overline{\mathbf{x}}_{12}^{\text{freq}} + \overline{\mathbf{x}}_3^{\text{freq}}. &(4.47)\end{aligned}$$

La méthode précédente est appliquée sur le mélange $\overline{\mathbf{x}}^{\text{freq}}$ afin d'estimer $\overline{\mathbf{x}}_{12}^{\text{freq}}$ et $\overline{\mathbf{x}}_3^{\text{freq}}$, puis la méthode est appliquée sur $\overline{\mathbf{x}}_{12}^{\text{freq}}$ pour estimer $\overline{\mathbf{x}}_1^{\text{freq}}$ et $\overline{\mathbf{x}}_2^{\text{freq}}$. Pour cela on considère les distributions :

$$p\left(\overline{\mathbf{x}}_{12}^{\text{freq}}\middle|\overline{\mathbf{x}}^{\text{freq}}\right) = \frac{p_{12}\left(\overline{\mathbf{x}}_{12}^{\text{freq}}\right)p_3\left(\overline{\mathbf{x}}^{\text{freq}}-\overline{\mathbf{x}}_{12}^{\text{freq}}\right)}{p\left(\overline{\mathbf{x}}^{\text{freq}}\right)} \xrightarrow{\mathcal{R}(\cdot)} \widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}, \widehat{\overline{\mathbf{x}}}_3^{\text{freq}}, \tag{4.48}$$

$$p\left(\overline{\mathbf{x}}_1^{\text{freq}}\middle|\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}\right) = \frac{p_1\left(\overline{\mathbf{x}}_1^{\text{freq}}\right)p_2\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}-\overline{\mathbf{x}}_1^{\text{freq}}\right)}{p_{12}\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}\right)} \xrightarrow{\mathcal{R}(\cdot)} \widehat{\overline{\mathbf{x}}}_1^{\text{freq}}, \widehat{\overline{\mathbf{x}}}_2^{\text{freq}}. \tag{4.49}$$

Les détails concernant le calcul de ces distributions sont disponibles en Annexe B.

Avantages de la proposition 1. Cette formalisation du problème à données manquantes pour la séparation à l'avantage de respecter le cahier des charges initial, notamment en terme de temps-réel. En effet, la solution finale étant analytique, il n'y aura pas besoin de faire appel à des algorithmes itératifs de type EM au moment de l'inférence, qui peuvent être trop long à converger par rapport à notre cas d'utilisation. De plus, le formalisme des modèles de mélanges (gaussien en l'occurrence) apporte une certaine flexibilité dans la modélisation des signaux. Enfin, le cas des lois normales amène des calculs relativement légers, tant en terme de calculs algébriques qu'en terme computationnel (en simulation). Une étude expérimentale de la variabilité de la distribution conditionnelle (4.40) est disponible dans la section expérimentale 4.4 et montre que la distribution estime les sources avec peu de variabilité en terme de scores objectifs.

Limitations de la proposition 1. Une illustration de l'évolution de cette distribution conditionnelle (4.40) est donnée en Figure 4.2. Un son de la base d'apprentissage a été sélectionné au hasard, puis la valeur de $p\left(\overline{\mathbf{x}}_1^{\text{freq}}\middle|\overline{\mathbf{x}}^{\text{freq}}\right)$ est calculée pour les frames de ce son (ligne bleue, pas de déformation). Une déformation aléatoire sur le spectre complexe est ensuite appliquée pour étudier l'influence d'un spectre éloigné de l'apprentissage sur la valeur de la probabilité. La déformation consiste en une perturbation aléatoire issue d'une loi uniforme sur chaque bin fréquentielle, de plus ou moins grande intensité (ligne rouge et orange, respectivement petit et grande déformation). On remarque que cette probabilité diminue pour des déformations importantes de ce spectre. Il est donc possible que la méthode ait des difficultés à généraliser

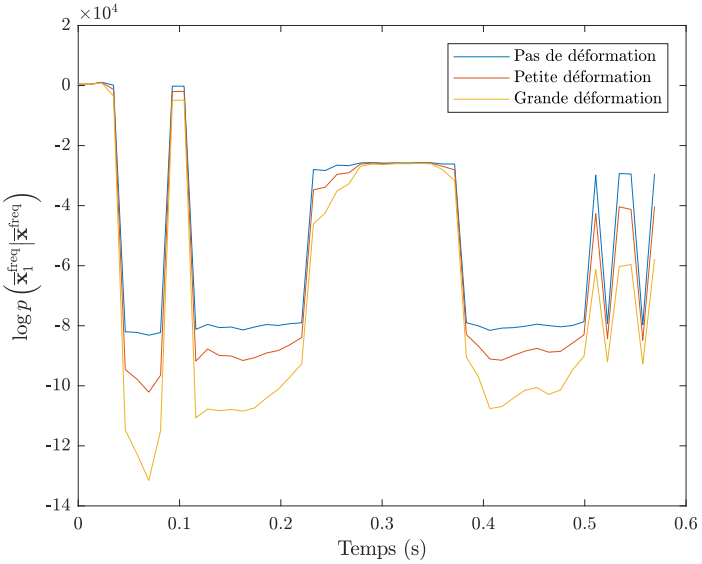


FIGURE 4.2 – Évolution de la probabilité conditionnelle en fonction du temps pour différentes déformations. Un son de la base d'apprentissage a été sélectionné au hasard, puis la valeur de $p\left(\overline{\mathbf{x}}_1^{\text{freq}} \middle| \overline{\mathbf{x}}^{\text{freq}}\right)$ est calculée pour les frames de ce son (ligne bleu, pas de déformation). Une déformation aléatoire sur le spectre complexe est ensuite appliquée pour étudier l'influence d'un spectre éloigné de l'apprentissage sur la valeur de la probabilité. La déformation consiste en une perturbation aléatoire issue d'une loi uniforme sur chaque bin fréquentielle, de plus ou moins grande intensité (ligne rouge et orange, respectivement petit et grande déformation).

lorsque les signaux sont différents de l'apprentissage. La proposition 2 prend en compte ce type de déformation dans sa modélisation et permet donc de pallier ce potentiel problème. De plus, l'hypothèse de stationnarité entre les frames (4.25) ne permet pas de modéliser les possibles transitions entre frames, contrairement à la proposition 2.

4.3.5 Proposition 2 : Estimation par déformation optimale (Def-MAP)

Formalisation. Dans cette proposition, l'idée est de chercher à déformer des spectres de l'ensemble d'apprentissage pour «coller» au mieux au spectre observé dans un contexte de modèle probabiliste. On note $\mathcal{S}_\tau = \left\{\mathbf{x}_{1\tau}^{\text{freq}}, \mathbf{x}_{2\tau}^{\text{freq}} : \mathbf{x}_\tau^{\text{freq}} = \mathbf{x}_{1\tau}^{\text{freq}} + \mathbf{x}_{2\tau}^{\text{freq}}\right\}$ la contrainte de somme à la frame τ, et on note $y_{z,\tau}$ l'indice du son de la classe z à la frame τ dans l'ensemble d'apprentissage. On suppose une distribution uniforme sur les indices de sons :

$$p(y_{z,\tau}) = \frac{1}{N_z} \tag{4.50}$$

avec N_z le nombre de sons dans la classe z. La transition des sons entre deux frames est également considérée. Elle est calculée avec la distance, à une frame donnée τ, entre $\mathbf{x}_{y_{z,\tau},z\tau}^{\text{freq}}$ correspondant au son $y_{z,\tau}$ à la frame τ et $\mathbf{x}_{y_{z,\tau-1}z\tau}^{\text{freq}}$ correspondant au son $y_{z,\tau-1}$ de la frame précédente $\tau-1$, ce qui s'écrit de la manière suivante :

$$p\left(y_{z,\tau}\middle|y_{z,\tau-1}\right) = \frac{1}{Z}\exp\left(-d\left(\mathbf{x}^{\text{freq}}_{y_{z,\tau},z\tau}, \mathbf{x}^{\text{freq}}_{y_{z,\tau-1},z\tau}\right)\right) \tag{4.51}$$

avec $d(\mathbf{a},\mathbf{b})$ une distance entre $\mathbf{a}$ et $\mathbf{b}$ et Z un coefficient normalisateur tel que la somme de ces probabilités soit égale à 1, $\sum_{y_{z,\tau}} p(y_{z,\tau}|y_{z,\tau-1}) = 1$. Ces probabilités de transitions sont calculées sur l'ensemble d'apprentissage avant toute phase de séparation pour toutes les classes z. On considère comme distance la corrélation de Pearson entre deux spectres en échelle dB. On note $\mathbf{x}_{\text{dB}} = 20\log_{10}(|\mathbf{x}^{\text{freq}}|)$. La distance s'écrit :

$$d(\mathbf{x}_{\text{dB}}, \mathbf{y}_{\text{dB}}) = -2\log_{10}\left(\frac{\mathbf{x}_{\text{dB}}^{\top}\mathbf{y}_{\text{dB}} - B\mathbb{E}[\mathbf{x}_{\text{dB}}]\mathbb{E}[\mathbf{y}_{\text{dB}}]}{(B-1)\sqrt{\mathbb{V}[\mathbf{x}_{\text{dB}}]\mathbb{V}[\mathbf{y}_{\text{dB}}]}}\right), \tag{4.52}$$

avec $\mathbb{E}[\mathbf{x}_{\text{dB}}]$ la moyenne, $\mathbb{V}[\mathbf{x}_{\text{dB}}]$ la variance et B la taille de $\mathbf{x}_{\text{dB}}$.

Traitement de la première frame ($\tau = 1$)

La séparation est réalisée de manière séquentielle. On se place dans le cas où $z_1 = 1$ et $z_2 = 2$ par soucis de notation. On cherche d'abord à séparer la première frame ($\tau = 1$), en modélisant la distribution des deux frames comme suit :

$$p\left(\mathbf{x}^{\text{freq}}_{11}, \mathbf{x}^{\text{freq}}_{21}\middle|\mathcal{S}_1\right) = \sum_{y_{11}}\sum_{y_{21}} p\left(\mathbf{x}^{\text{freq}}_{11}, \mathbf{x}^{\text{freq}}_{21}\middle|y_{11}, y_{21}, \mathcal{S}_1\right) p\left(y_{11}, y_{21}\middle|\mathcal{S}_1\right). \tag{4.53}$$

Cette distribution signifie que l'on cherche les couples de spectres les plus probables connaissant le mélange. De part le modèle génératif, on a les propriétés d'indépendances suivantes :

$$p\left(y_{11}, y_{21}\middle|\mathcal{S}_1\right) = p(y_{11}, y_{21}) \tag{4.54}$$

$$p(y_{11}, y_{21}) = p(y_{11})\,p(y_{21}). \tag{4.55}$$

Cela permet de simplifier la distribution :

$$p\left(\mathbf{x}^{\text{freq}}_{11}, \mathbf{x}^{\text{freq}}_{21}\middle|\mathcal{S}_1\right) = \sum_{y_{11}}\sum_{y_{21}} p\left(\mathbf{x}^{\text{freq}}_{11}, \mathbf{x}^{\text{freq}}_{21}\middle|y_{11}, y_{21}, \mathcal{S}_1\right) p(y_{11})\,p(y_{21}). \tag{4.56}$$

L'idée de la proposition est de chercher une estimation des spectres complexes en déformant des spectres de l'ensemble d'apprentissage.

Définition 22 (Transformation des spectres)**.** *La transformation, paramétrée par $\mathbf{T}_{y_{z1}z1}$, est définie comme une régression de la forme :*

$$\mathbf{x}^{freq}_{z1} = \mathbf{T}_{y_{z1}z1} \cdot \mathbf{x}^{freq}_{y_{z1}z1} + \epsilon_{z1} \tag{4.57}$$

où ϵ_{z1} est un bruit additif.

En effet la transformation va dépendre à la fois de la classe et de l'indice du son. L'objectif est de trouver la transformation *optimale* de $\mathbf{x}^{\text{freq}}_{y_{z1}z1}$ par $\mathbf{T}_{y_{z1}z1}$ telle que la somme soit le mélange observé, c'est-à-dire le problème d'optimisation suivant :

$$\min_{\mathbf{T}_{y_{11}11}, \mathbf{T}_{y_{21}21}} \mathcal{L}\left(\mathbf{T}_{y_{11}11}\right) + \mathcal{L}\left(\mathbf{T}_{y_{21}21}\right) \tag{4.58}$$

$$\text{s.c. } \mathbf{T}_{y_{11}11} \cdot \mathbf{x}^{\text{freq}}_{y_{11}11} + \mathbf{T}_{y_{21}21} \cdot \mathbf{x}^{\text{freq}}_{y_{21}21} = \mathbf{x}_1 \tag{4.59}$$

avec $\mathcal{L}(\cdot)$ une fonction de coût à minimiser. Cette transformation peut être, notamment :

— Variation de l'amplitude (locale ou globale) d'un facteur $\mathbf{V} = \left(V_f\right)_{f=1}^{B}$:

$$\mathbf{T}^{(\mathbf{V})}\left(\mathbf{x}^{\text{freq}}\right) = \mathbf{V} \cdot \left|\mathbf{x}^{\text{freq}}\right| \cdot e^{i \angle \mathbf{x}^{\text{freq}}} \tag{4.60}$$

— Déphasage (local ou global) d'un facteur $\boldsymbol{\phi} = \left(\phi_f\right)_{f=1}^{B}$:

$$\mathbf{T}^{(\boldsymbol{\phi})}\left(\mathbf{x}^{\text{freq}}\right) = \left|\mathbf{x}^{\text{freq}}\right| \cdot e^{i\left(\angle \mathbf{x}^{\text{freq}} + \boldsymbol{\phi}\right)}. \tag{4.61}$$

Des transformations similaires sont également possibles dans le domaine temporel. La transformation globale peut être une combinaison de ces transformations :

$$\mathbf{T} = \mathbf{T}^{(\mathbf{V})} \circ \mathbf{T}^{(\boldsymbol{\phi})}. \tag{4.62}$$

Remarque. Cette phase d'optimisation permettra de dériver la loi jointe (4.78) des deux spectres à déformer en convertissant la fonction de perte de cette optimisation en probabilité.

Transformation d'amplitude. Dans un premier temps on considère une modification de l'amplitude, et la phase du mélange sera ajoutée au moment de la reconstruction. Pour simplifier la présentation on ne garde que l'indice de chaque classe et on retire l'indice de la frame et du son. Le problème d'optimisation devient donc :

$$\min_{\mathbf{T}_1, \mathbf{T}_2 \in \mathbf{R}^B} \|\mathbf{T}_1 - \mathbf{1}\|^2 + \|\mathbf{T}_2 - \mathbf{1}\|^2 \tag{4.63}$$

$$\text{s.c. } \mathbf{T}_1 \cdot \left|\mathbf{x}_1^{\text{freq}}\right| + \mathbf{T}_2 \cdot \left|\mathbf{x}_2^{\text{freq}}\right| = \left|\mathbf{x}^{\text{freq}}\right|. \tag{4.64}$$

Ici la fonction à minimiser $\mathcal{L}(\cdot)$ est le carré de la norme de la différence entre le paramètre de variation de l'amplitude et le vecteur identité $\mathbf{1}$. En effet l'idée est de déformer les spectres présents dans la base d'apprentissage mais en gardant les spectres déformés les plus proches possible des originaux. Les détails de calculs pour la résolution de ce problème d'optimisation sont disponibles dans l'Annexe B.4 et le résultat est :

$$\widehat{\mathbf{T}}_1 = \frac{\left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_1^{\text{freq}}\right| \cdot \left(\left|\mathbf{x}^{\text{freq}}\right| - \left|\mathbf{x}_2^{\text{freq}}\right|\right)}{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}} \tag{4.65}$$

$$\widehat{\mathbf{T}}_2 = \frac{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right| \cdot \left(\left|\mathbf{x}^{\text{freq}}\right| - \left|\mathbf{x}_1^{\text{freq}}\right|\right)}{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}}, \tag{4.66}$$

avec $\cdot^{2}$ la puissance élément par élément. L'estimation de l'amplitude des spectres est donc :

$$\widehat{\left|\mathbf{x}_1^{\text{freq}}\right|} = \left|\mathbf{x}^{\text{freq}}\right| \cdot \frac{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2}}{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}} + \frac{\left|\mathbf{x}_1^{\text{freq}}\right| \cdot \left|\mathbf{x}_2^{\text{freq}}\right| \cdot \left(\left|\mathbf{x}_2^{\text{freq}}\right| - \left|\mathbf{x}_1^{\text{freq}}\right|\right)}{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}} \tag{4.67}$$

$$\widehat{\left|\mathbf{x}_2^{\text{freq}}\right|} = \left|\mathbf{x}^{\text{freq}}\right| \cdot \frac{\left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}}{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}} + \frac{\left|\mathbf{x}_1^{\text{freq}}\right| \cdot \left|\mathbf{x}_2^{\text{freq}}\right| \cdot \left(\left|\mathbf{x}_1^{\text{freq}}\right| - \left|\mathbf{x}_2^{\text{freq}}\right|\right)}{\left|\mathbf{x}_1^{\text{freq}}\right|^{\cdot 2} + \left|\mathbf{x}_2^{\text{freq}}\right|^{\cdot 2}}. \tag{4.68}$$

Il est intéressant de noter que le problème posé plus haut n'a qu'une solution unique et analytique.

Cette propriété est très utile dans un contexte temps-réel où un solveur numérique et itératif serait beaucoup trop long pour notre cas d'utilisation. L'estimation de l'amplitude des spectres fait apparaître deux termes :

— Le premier peut être vu comme un masque de ratio d'énergie sur le spectre mélange :

$$\mathbf{M}_z = \frac{|\mathbf{x}_z^{\text{freq}}|^{\cdot 2}}{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_2^{\text{freq}}|^{\cdot 2}} \tag{4.69}$$

On peut le comparer au filtre de Wiener utilisé classiquement en séparation de sources comme l'estimateur des moindres carrées pour une source connaissant le mélange [48].

— Le second est en quelque sorte une correction apportée à l'estimation pour que la transformation soit bien l'identité lorsque l'amplitude du mélange correspond exactement aux deux amplitudes des spectres de l'apprentissage. En effet si $|\mathbf{x}^{\text{freq}}| = |\mathbf{x}_1^{\text{freq}}| + |\mathbf{x}_2^{\text{freq}}|$, alors on a :

$$\begin{aligned}
\widehat{|\mathbf{x}_1^{\text{freq}}|} &= |\mathbf{x}^{\text{freq}}| \cdot \frac{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2}}{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_2^{\text{freq}}|^{\cdot 2}} + \frac{|\mathbf{x}_1^{\text{freq}}| \cdot |\mathbf{x}_2^{\text{freq}}| \cdot \left(|\mathbf{x}_2^{\text{freq}}| - |\mathbf{x}_1^{\text{freq}}|\right)}{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_2^{\text{freq}}|^{\cdot 2}} && (4.70)\\
&= \left(|\mathbf{x}_1^{\text{freq}}| + |\mathbf{x}_2^{\text{freq}}|\right) \cdot \frac{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2}}{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_2^{\text{freq}}|^{\cdot 2}} + \frac{|\mathbf{x}_1^{\text{freq}}| \cdot |\mathbf{x}_2^{\text{freq}}| \cdot \left(|\mathbf{x}_2^{\text{freq}}| - |\mathbf{x}_1^{\text{freq}}|\right)}{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_2^{\text{freq}}|^{\cdot 2}} && (4.71)\\
&= \frac{|\mathbf{x}_1^{\text{freq}}|^{\cdot 3} + |\mathbf{x}_2^{\text{freq}}| \cdot |\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_1^{\text{freq}}| \cdot |\mathbf{x}_2^{\text{freq}}|^{\cdot 2} - |\mathbf{x}_1^{\text{freq}}|^{\cdot 2} \cdot |\mathbf{x}_2^{\text{freq}}|}{|\mathbf{x}_1^{\text{freq}}|^{\cdot 2} + |\mathbf{x}_2^{\text{freq}}|^{\cdot 2}} && (4.72)\\
&= |\mathbf{x}_1^{\text{freq}}|. && (4.73)
\end{aligned}$$

Cette transformation en amplitude est souvent la plus utilisée dans la littérature mais sous une autre forme (via l'utilisation d'un masque). Or la phase n'est pas estimée dans ce cas et cela peut être problématique pour certains types de sons (comme les sons de type impact). C'est pourquoi on considère dans la suite une transformation complexe qui s'applique sur le spectre complexe directement : ainsi à la fois l'amplitude et la phase sont modifiées.

Transformation complexe. On cherche une transformation complexe via le problème d'optimisation suivant :

$$\begin{aligned}
&\min_{\mathbf{T}_1, \mathbf{T}_2 \in \mathbb{C}^B} \|\mathbf{T}_1 - \mathbf{1}\|^2 + \|\mathbf{T}_2 - \mathbf{1}\|^2 && (4.74)\\
&\text{s.c. } \mathbf{T}_1 \cdot \mathbf{x}_1^{\text{freq}} + \mathbf{T}_2 \cdot \mathbf{x}_2^{\text{freq}} = \mathbf{x}^{\text{freq}}. && (4.75)
\end{aligned}$$

Ici ce sont bien les spectres complexes qui sont utilisés et non les spectres d'amplitude. On transforme ce problème à variables complexes en problème à variables réelles en vectorisant les transformations et les spectres (comme dans la partie sur les données manquantes) : $\overline{\mathbf{T}} = [\text{Re}(\mathbf{T}); \text{Im}(\mathbf{T})] \in \mathbb{R}^{2B}$, $\overline{\mathbf{1}} = [\mathbf{1}_B; \mathbf{0}_B]$ et $\overline{\mathbf{x}}^{\text{freq}} = [\text{Re}(\mathbf{x}^{\text{freq}}); \text{Im}(\mathbf{x}^{\text{freq}})] \in \mathbb{R}^{2B}$. Ainsi on se ramène à une optimisation de vecteurs réels. De la même manière que pour la transformation d'amplitude, les détails de la résolution de ce problème d'optimisation sont disponibles en Annexe B.5. La solution est :

$$\widehat{\mathbf{T}}_1 = \frac{\mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right) \cdot \mathrm{Re}\left(\mathbf{x}^{\mathrm{freq}} - \mathbf{x}_2^{\mathrm{freq}}\right)}{\mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right)^{\cdot 2}} + i \frac{\mathrm{Im}\left(\mathbf{x}^{\mathrm{freq}}\right) \cdot \mathrm{Im}\left(\mathbf{x}_1^{\mathrm{freq}}\right)}{\mathrm{Im}\left(\overline{\mathbf{x}}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Im}\left(\overline{\mathbf{x}}_2^{\mathrm{freq}}\right)^{\cdot 2}}, \tag{4.76}$$

$$\widehat{\mathbf{T}}_2 = \frac{\mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right) \cdot \mathrm{Re}\left(\mathbf{x}^{\mathrm{freq}} - \mathbf{x}_1^{\mathrm{freq}}\right)}{\mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right)^{\cdot 2}} + i \frac{\mathrm{Im}\left(\mathbf{x}^{\mathrm{freq}}\right) \cdot \mathrm{Im}\left(\mathbf{x}_2^{\mathrm{freq}}\right)}{\mathrm{Im}\left(\overline{\mathbf{x}}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Im}\left(\overline{\mathbf{x}}_2^{\mathrm{freq}}\right)^{\cdot 2}}. \tag{4.77}$$

Retour aux distributions. Une fois la transformation estimée, on peut s'en servir pour calculer la distribution conditionnelle en convertissant la fonction de coût de l'optimisation en probabilité via la formule suivante :

$$p\left(\mathbf{x}_{11}^{\mathrm{freq}}, \mathbf{x}_{21}^{\mathrm{freq}} \middle| y_{11}, y_{21}, \mathcal{S}_1\right) \propto \exp\left(-\alpha\left(\mathcal{L}\left(\widehat{\mathbf{T}}_{y_{11}1}1\right) + \mathcal{L}\left(\widehat{\mathbf{T}}_{y_{21}2}1\right)\right)\right). \tag{4.78}$$

avec α un coefficient régularisateur permettant de régler «l'attache» à l'ensemble d'apprentissage. Ce coefficient peut être réglé sur un ensemble de développement ou fixé par avance. Les frames modifiées sont ensuite sélectionnées suivant le MAP de $p\left(\mathbf{x}_{11}^{\mathrm{freq}}, \mathbf{x}_{21}^{\mathrm{freq}} \middle| y_{11}, y_{21}, \mathcal{S}_1\right)$.

Cette proposition 2 pour la séparation réalise ainsi une régression non paramétrique où l'estimation d'une frame déformée consiste en l'espérance de l'expression définie en Eq. (4.57).

Traitement de la deuxième frame ($\tau = 2$)

Pour la deuxième frame, on procède d'une manière similaire en considérant tout d'abord la distribution conditionnelle des frames à séparer sachant les précédentes frames :

$$p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| \mathbf{x}_{11}^{\mathrm{freq}}, \mathbf{x}_{21}^{\mathrm{freq}}, \mathcal{S}_2\right) = p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| y_{11}, y_{21}, \mathcal{S}_2\right) \tag{4.79}$$

$$= \sum_{y_{12}} \sum_{y_{22}} p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| y_{12}, y_{22}, \mathcal{S}_2\right) p\left(y_{12}, y_{22} \middle| y_{11}, y_{21}, \mathcal{S}_2\right). \tag{4.80}$$

De même, grâce au modèle génératif on a les hypothèses d'indépendance suivantes :

$$p\left(y_{12}, y_{22} \middle| y_{11}, y_{21}, \mathcal{S}_2\right) = p\left(y_{12}, y_{22} \middle| y_{11}, y_{21}\right) \tag{4.81}$$

$$p\left(y_{12}, y_{22} \middle| y_{11}, y_{21}\right) = p\left(y_{12} \middle| y_{11}\right) p\left(y_{22} \middle| y_{21}\right). \tag{4.82}$$

On peut donc simplifier la distribution conditionnelle :

$$p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| y_{11}, y_{21}, \mathcal{S}_2\right) = \sum_{y_{12}} \sum_{y_{22}} p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| y_{12}, y_{22}, \mathcal{S}_2\right) p\left(y_{12} \middle| y_{11}\right) p\left(y_{22} \middle| y_{21}\right). \tag{4.83}$$

Ensuite on marginalise sur les frames précédentes pour obtenir la distribution jointe :

$$p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| \mathcal{S}_2\right) = \sum_{y_{11}} \sum_{y_{21}} p\left(\mathbf{x}_{12}^{\mathrm{freq}}, \mathbf{x}_{22}^{\mathrm{freq}} \middle| y_{11}, y_{21}, \mathcal{S}_2\right) p\left(y_{11}\right) p\left(y_{21}\right). \tag{4.84}$$

Connaissant les indices de la frame 1, on peut estimer les indices de la frame 2 et ainsi résoudre le problème d'optimisation précédent pour trouver les spectres de la frame 2. Ce procédé est généralisé au N frames pour obtenir une estimation de chaque spectre.

Reconstruction des sources. Grâce à la distribution conditionnelle précédente, on peut estimer les spectres complexes en sélectionnant les deux spectres déformés les plus probables au sens du MAP, c'est-à-dire :

$$\widehat{\mathbf{x}}_{1\tau}^{\text{freq}}, \widehat{\mathbf{x}}_{2\tau}^{\text{freq}} = \underset{\mathbf{x}_{1\tau}^{\text{freq}}, \mathbf{x}_{2\tau}^{\text{freq}}}{\operatorname{argmax}} p\left(\mathbf{x}_{1\tau}^{\text{freq}}, \mathbf{x}_{2\tau}^{\text{freq}} \middle| y_{1\tau}, y_{2\tau}, \mathcal{S}_\tau\right). \tag{4.85}$$

La fonction de séparation est ici le MAP des distributions conditionnelles à chaque frame :

$$\widehat{\mathcal{R}}\left(\mathbf{x}_1^{\text{freq}}, ..., \mathbf{x}_N^{\text{freq}}\right) = \left(\underset{\mathbf{x}_{11}^{\text{freq}}, \cdot}{\operatorname{argmax}} p\left(\mathbf{x}_{11}^{\text{freq}}, \cdot \middle| y_{11}, y_{21}, \mathcal{S}_1\right), ..., \underset{\mathbf{x}_{1N}^{\text{freq}}, \cdot}{\operatorname{argmax}} p\left(\mathbf{x}_{1N}^{\text{freq}}, \cdot \middle| y_{1N}, y_{2N}, \mathcal{S}_N\right)\right). \tag{4.86}$$

4.3.6 Analyse des propositions

Les deux propositions précédentes présentent chacune des avantages et des inconvénients. La proposition 1 a l'avantage d'être relativement simple à mettre en œuvre et permet rapidement de réaliser un compromis entre les performances de séparation et le temps de calcul en sélectionnant le nombre de composantes des mélanges de manière adéquate. Néanmoins l'inconvénient principal est que les transitions temporelles entre les frames ne sont pas prises en compte. Une solution pourrait être de combiner le GMM avec un HMM comme dans [10]. C'est pour palier à ce problème que la proposition 2 a été proposée. En plus de modéliser les transitions entre les frames, cette proposition permet également une interprétabilité de la séparation grâce à l'utilisation de sons connus que l'on cherche à déformer. En effet, les transformations considérées sont relativement communes dans le domaine du traitement du signal (modification de l'amplitude et de la phase). De plus la résolution du problème d'optimisation permet de quantifier «l'effort» de déformation qu'il a fallu apporter pour chaque élément de l'ensemble d'apprentissage. Néanmoins sa formulation actuelle est d'une trop grande complexité algorithmique pour être utilisée dans des temps raisonnables vis-à-vis du temps-réel.

La proposition 1 peut être utilisé dans un contexte temps-réel sous condition que les lois mélanges ne comportent pas beaucoup de composantes (voir les expériences). La proposition 2 quant à elle peut difficilement être utilisée en temps-réel car le nombre de prototypes à tester pour les déformations est trop important.

Les résultats expérimentaux (voir partie 4.4) suggèrent que la proposition 1 (DM-GMM) est meilleure que la proposition 2 (Def-MAP). Néanmoins il est prématuré de porter une décision définitive sur le meilleur choix à faire ici. En effet, la proposition 2 n'est pas exploitée au maximum de ces capacités en l'état car il manque une étape de réduction de données (de la même manière que pour la classification).

4.4 Expériences numériques sur données réelles.

Dans cette partie nous allons présenter les bases de données utilisées par la communauté de la séparation de sources et celles utilisées dans les expériences numériques, ainsi que les métriques usuelles pour attester de la performance des méthodes de séparation. De plus les résultats de notre méthode ainsi que de quelques méthodes concurrentes sont également présentés.

4.4.1 Ensembles données et métriques pour la séparation de sources

Bases de données. Les bases de données publiques pour la séparation de sources sont surtout focalisées sur la séparation des instruments de musique ainsi que la séparation entre voix et bruits de fond. La base de données DEMAND [111] contient plusieurs contextes acoustiques (Domestique, Nature, Bureau, Public, Rue, Transport) représentant des bruits de fonds relatifs à ces contextes. Concernant la séparation d'instruments de musique, deux bases sont principalement utilisées. La première est DSD100 [61] qui contient 100 enregistrements de musiques dont les mélanges et les instruments séparés (voix, basse, batterie et autres) sont disponibles. La base MUSDB2018 [89] contient 150 morceaux de musiques et a été utilisée notamment pour le concours Sisec2018 pour la séparation d'instruments de musique.

Dans nos expériences, nous utiliserons aussi la base de données A-Volute (déjà utilisée en classification), notamment les classes voix et détonation. Des mélanges de ces sons seront créés de manière artificielle pour vérifier la pertinence de notre méthode de séparation.

Métriques. Les métriques d'évaluation des méthodes de séparation ont été proposées au départ par Vincent *et al.* [118] puis ont été implémentées dans une librairie appelée BSS Eval. Elles sont encore largement utilisées de nos jours car elle permettent une évaluation objective des méthodes, au contraire de la métrique PEASS de Emiya *et al.* [31].

Les sources estimées sont décomposées de la manière suivante :

$$\widehat{\mathbf{s}}_j = \mathbf{s}_{\text{target}} + \mathbf{e}_{\text{interf}} + \mathbf{e}_{\text{noise}} + \mathbf{e}_{\text{artif}}. \tag{4.87}$$

On définit les opérateurs de projections suivant :

— $\Pi_{\mathbf{s}_j}$ la projection sur le sous-espace engendré par $\mathbf{s}_j$,
— $\Pi_{\mathbf{s}}$ la projection sur le sous-espace engendré par $\mathbf{s}_1, ..., \mathbf{s}_n$,
— $\Pi_{\mathbf{s},\mathbf{n}}$ la projection sur le sous-espace engendré par $\mathbf{s}_1, ..., \mathbf{s}_n, \mathbf{n}_1, ..., \mathbf{n}_m$.

Ainsi les signaux de la décomposition (4.87) sont définis par :

$$\begin{aligned}
\mathbf{s}_{\text{target}} &= \Pi_{\mathbf{s}_j}\widehat{\mathbf{s}}_j, && (4.88)\\
\mathbf{e}_{\text{interf}} &= \Pi_{\mathbf{s}}\widehat{\mathbf{s}}_j - \Pi_{\mathbf{s}_j}\widehat{\mathbf{s}}_j, && (4.89)\\
\mathbf{e}_{\text{noise}} &= \Pi_{\mathbf{s},\mathbf{n}}\widehat{\mathbf{s}}_j - \Pi_{\mathbf{s}}\widehat{\mathbf{s}}_j, && (4.90)\\
\mathbf{e}_{\text{artif}} &= \widehat{\mathbf{s}}_j - \Pi_{\mathbf{s},\mathbf{n}}\widehat{\mathbf{s}}_j. && (4.91)
\end{aligned}$$

Basées sur cette décomposition, les métriques de performance pour la séparation de sources sont au nombre de quatre :

— Le rapport signal à distorsion (SDR), défini par :

$$\text{SDR} = 10\log_{10}\left(\frac{\|\mathbf{s}_{\text{target}}\|^2}{\|\mathbf{e}_{\text{interf}} + \mathbf{e}_{\text{noise}} + \mathbf{e}_{\text{artif}}\|^2}\right), \tag{4.92}$$

— Le rapport signal à interférence (SIR), défini par :

$$\text{SIR} = 10\log_{10}\left(\frac{\|\mathbf{s}_{\text{target}}\|^2}{\|\mathbf{e}_{\text{interf}}\|^2}\right), \tag{4.93}$$

— Le rapport signal à bruit (SNR), défini par :

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\|\mathbf{s}_{\mathrm{target}} + \mathbf{e}_{\mathrm{interf}}\|^2}{\|\mathbf{e}_{\mathrm{noise}}\|^2}\right), \tag{4.94}$$

— Le rapport signal à interférence (SAR), défini par :

$$\mathrm{SAR} = 10\log_{10}\left(\frac{\|\mathbf{s}_{\mathrm{target}} + \mathbf{e}_{\mathrm{interf}} + \mathbf{e}_{\mathrm{noise}}\|^2}{\|\mathbf{e}_{\mathrm{artif}}\|^2}\right). \tag{4.95}$$

Ces métriques représentent des ratios de performances énergétiques en terme de suppression de sources «parasites» dans le signal voulu (artéfacts, interférences, distorsions ou bruits). Vu la définition de ces métriques, plus le score sera élevé meilleur sera le résultat. A titre d'illustration, on comparera le score SDR de la méthode considérée à un oracle (IRM par exemple), et si le score SDR de la méthode considérée est proche de l'oracle, la méthode pourra être considérée comme très bonne. De plus, si le score est négatif, la méthode sera considérée mauvaise. En effet, de part leur définition, ces métriques peuvent prendre des valeurs entre $-\infty$ et $+\infty$. Ainsi, la valeur minimale de référence sera par exemple celle de l'oracle MIX (équivalent du «tirage au hasard» de la classification), et la valeur maximale de référence sera par exemple celle de l'oracle IRM (ou IBM).

Remarque. Ces métriques sont valables uniquement dans le cas où seule une distorsion de type gain invariant dans le temps est considérée. Pour les autres cas, il faut modifier les opérateurs de projections pour prendre en compte des filtrages ou des variations dans le temps (voir [118] pour plus de détails).

Remarque : pertinence des métriques objectives. De plus en plus d'auteurs mettent l'accent dans leur article sur l'utilisation de métriques subjectives, même si les métriques objectives restent le juge de paix pour attester des performances de la séparation. En effet, il a été observé que le lien fin entre scores objectifs et perceptifs dans une gamme de scores limitée n'est pas évident. Les tests subjectifs consistent à demander à un panel d'auditeurs de noter les résultats de séparation en comparaison avec le vrai signal sous deux aspects :

— *Y a-t-il des artéfacts ?* Les interférences sont du contenu spectral créé artificiellement et qui ne vient pas d'une source.

— *Y a-t-il des interférences ?* Les artéfacts sont à comprendre au sens de sources parasites dans les estimations : par exemple dans l'estimation guitare est présente une partie de la voix.

Par manque de temps et de panel d'auditeurs, seules les métriques objectives seront utilisées dans nos expériences.

4.4.2 Design des expériences

Réglages des paramètres. Les sons sont rééchantillonnés à $F_e = 44,1$kHz. La taille de la fenêtre d'analyse est réglée à $\Delta t = 1024$ (23,2ms) et le décalage à $\delta t = 512$ (11,6ms), et la fenêtre d'analyse est la racine de la fenêtre de Hanning. Le nombre de fréquences gardées dans le spectre est fixé à $B = \Delta t/2 + 1$ (la première moitié du spectre jusque la fréquence de Nyquist). Les sons de tests sont des portions de sons mélangés d'une durée $\Delta T = 1$s, c'est-à-dire $N = 84$ frames. Pour la création du mélange, on choisit aléatoirement dans deux classes des sons découpés que l'on mélange en proportion 50/50. Les résultats pour la partie déformation optimale seront divisés suivant les résultats sur l'ensemble d'apprentissage et les résultats sur l'ensemble de test.

Méthodes testées. La méthode RASE est utilisée à la fois avec les données manquantes (DM-GMM) pour trois jeux de paramètres (10, 50 et 100 composantes par classe) et les déformations en sélectionnant la meilleure déformation sur la base du MAP de la probabilité jointe (Def-MAP) pour 3 valeurs de $\alpha = 0,01,\ 0,1,\ 1$. La méthode RARE est également considérée pour la séparation, utilisée comme suit : on cherche dans l'ensemble d'apprentissage les spectres les plus proches du mélange au sens de la vraisemblance, puis on déforme les deux spectres en amplitude. La méthode NMF supervisée est considérée, et l'apprentissage est fait avec un nombre de bases spectrales de 100 dans chaque classe, de même que la méthode PLCA supervisée. Les méthodes NMF et PLCA non supervisées sont utilisées comme suit : elles cherchent 2 sources (donc 2 composantes) en 300 itérations sur un ensemble de 40 frames glissantes. On considère enfin un DNN appelé MSonyNet qui correspond à une architecture type MMDenseNet mais avec des blocs LSTM au lieu de DenseBloc. Les méthodes oracles sont également considérés.

4.4.3 Résultats de la séparation sur l'ensemble d'apprentissage

Le Tableau 4.1 présente les résultats des métriques objectives (SDR) pour la séparation de voix et de détonation sur l'ensemble d'apprentissage pour les méthodes oracles. L'oracle MIX donne normalement la «pire» estimation au sens où elle ne fournit aucune information sur les sources. Les oracles IBM et IRM donnent des bornes supérieures des scores et peuvent être considérées comme les meilleures performances atteignables. Les valeurs minimum et maximum de ces métriques seront considérées comme les bornes inférieures et supérieures pour comparer aux autres méthodes.

Le Tableau 4.2 présente les résultats pour notre méthode de séparation et les méthodes concurrentes en terme de SDR. La méthode RASE par déformation optimale (Def-MAP) présente des résultats très convenables car sur l'ensemble d'apprentissage elle retrouvera toujours les bons spectres mélangés étant donné que la fonction objectif de l'optimisation sera égale à 0 pour ces spectres-là. Le coefficient α dans la distribution (4.78) a bien un rôle régularisateur car les performances sur l'ensemble d'apprentissage se dégradent en fonction du coefficient. De même, la proposition 1 (DM-GMM) est assez satisfaisante bien que légèrement moins performante que le MSonyNet et bien moins que la proposition 2. Plus le nombre de composantes est élevé, meilleurs sont les performances. En effet les distributions des sources individuelles auront un plus grand pouvoir de représentation avec beaucoup de composantes.

La méthode RARE, bien que naïvement appliquée dans ce contexte, semble donner des résultats similaires aux méthodes supervisées concurrentes (NMF et PLCA), au moins pour la source Voix. Cela suggère que la classification des frames de Voix est plus efficace que pour les frames de Détonation.

Les méthodes non supervisées ne donnent pas de très bons résultats car elles ne sont pas faites pour fonctionner en temps-réel et ont le principal problème de la permutation des sources. Ce problème est récurrent dans le cas de la séparation non supervisée à la frame. Supposons qu'à une frame donnée, l'estimation des sources se fait dans l'ordre 1 - 2. À la frame suivante, il est autant probable de prédire les sources dans l'ordre 1 - 2 que dans l'ordre 2 - 1. Cela constitue le problème de la permutation des sources.

Les scores SIR et SAR sont disponibles dans l'Annexe C.

4.4.4 Résultats de la séparation sur l'ensemble test

Le Tableau 4.3 présente les résultats des métriques objectives (SDR) pour la séparation de voix et de détonation sur l'ensemble de test pour les méthodes oracles. De la même manière que

Tableau 4.1 – Tâche de séparation sur l'ensemble d'apprentissage. Score SDR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les oracles. Plus le score SDR est élevé meilleur est le résultat.

Méthode	Param.	SDR	
		Voix	Détonation
MIX	-	4,6 (5,5)	-4,2 (5,3)
IBM	1	14,2 (3,6)	9,3 (3,6)
	2	14,5 (3,4)	9,5 (3,6)
IRM	1	14,8 (3,6)	9,8 (3,7)
	2	15,3 (3,6)	10,6 (3,8)

Tableau 4.2 – Tâche de séparation sur l'ensemble d'apprentissage. Score SDR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les différentes méthodes : la méthode RASE (déclinée suivant les deux propositions DM-GMM et Def-MAP respectivement), la méthode RARE, ainsi que les méthodes concurrentes NMF et PLCA (non supervisée et supervisée respectivement) et le MSonyNet (réseau de neurones). Plus le score SDR est élevé meilleur est le résultat.

Méthode		Param.	SDR	
			Voix	Détonation
			Min. 4,6 Max. 15,3	Min. -4,2 Max. 10,6
RASE	DM-GMM	10 comp.	8,1 (4,6)	2,4 (4,6)
		50 comp.	8,4 (4,5)	2,0 (4,9)
		100 comp.	8,7 (4,7)	2,5 (4,9)
RASE	Def-MAP	$\alpha = 10^{-4}$	6,0 (3,2)	-1,5 (5,2)
		$\alpha = 10^{-3}$	7,2 (3,7)	0,3 (5,2)
		$\alpha = 10^{-2}$	12,4 (4,3)	6,9 (4,7)
		$\alpha = 10^{-1}$	13,4 (3,7)	8,5 (4,1)
		$\alpha = 1$	13,4 (3,6)	8,4 (4,1)
		$\alpha = 10$	13,4 (3,6)	8,4 (4,1)
RARE	-	-	7,2 (3,4)	0.3 (5,2)
NMF	Non sup	2 comp.	3,3 (3,4)	-5,3 (5,7)
	Sup	100 comp.	7,9 (4,7)	3,9 (4,1)
PLCA	Non sup	2 comp.	2,6 (3,7)	-5,5 (5,5)
	Sup	100 comp.	8,1 (4,7)	2,3 (4,6)
MSonyNet	-	-	9,4 (4,6)	4,7 (5,3)

Tableau 4.3 – Tâche de séparation sur l'ensemble de test. Score SDR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les oracles. Plus le score SDR est élevé meilleur est le résultat.

Méthode	Param.	SDR	
		Voix	Détonation
MIX	-	4,9 (4,2)	-4,6 (4,2)
IBM	1	15,1 (3,2)	9,0 (3,0)
	2	15,4 (3,2)	9,3 (3,0)
IRM	1	15,8 (3,3)	9,7 (3,1)
	2	16,5 (3,4)	10,4 (3,2)

précédemment, les oracles MIX et IBM/IRM donnent respectivement les «pires» et «meilleures» performances possibles.

Le Tableau 4.4 présente les résultats pour notre méthode de séparation et les méthodes concurrentes en terme de SDR. Les résultats pour la méthode RASE (Def-MAP) diminuent grandement par rapport à l'ensemble d'apprentissage. Ce résultat était attendu car la méthode est dépendante de le frame τ où est réalisé la séparation, or la synchronisation des sons peut être différente entre l'apprentissage et le test. De plus comme la transformation est supposée être proche de l'identité, on ne peut pas transformer les sons de manière drastique. De plus, l'effet du coefficient α ne permet pas de contre-balancer cette attache trop forte à l'ensemble d'apprentissage dans ce cas là. En revanche la proposition 1 (DM-GMM) obtient d'assez bons résultats en comparaison des autres méthodes.

Les méthodes concurrentes ont des résultats similaires à ceux sur l'ensemble d'apprentissage, en dehors de RARE qui perd en performance.

En terme de temps de calcul, peu de méthodes parviennent à réaliser la séparation en temps-réel, notamment la proposition 2 (Def-MAP) : en effet cette méthode doit calculer des déformations suivant toutes les combinaisons de spectres dans l'ensemble d'apprentissage, ce qui représente un nombre conséquent dans notre cas (10k combinaisons). La proposition 1 quant à elle permet de sélectionner un compromis performance - temps de calcul en jouant sur le nombre de composantes des mélanges gaussiens.

Les scores SIR et SAR sont disponibles dans l'Annexe C.

4.4.5 Variabilité de la distribution conditionnelle de la proposition 1

Concernant la proposition 1, l'estimation des spectres se fait avec l'espérance conditionnelle de la distribution conditionnelle. Pour tester la variabilité de cette distribution, le protocole suivant est mis en place. Un mélange particulier est sélectionné, puis 10k échantillons sont générés à partir de la distribution conditionnelle. Ces échantillons permettent de recréer des estimations «aléatoires» des sources présentes dans le mélange. Le SDR pour chacune de ces estimations est calculé.

Les résultats de cette expérience sont regroupés dans le Tableau 4.5. Les scores moyens ne sont pas très bons car les estimations sont construites à partir de réalisations aléatoires de la distribution conditionnelle (4.40), ce qui était attendu. En revanche, l'écart-type sur les différents scores est très faible comparé à l'écart-type des scores de la procédure standard. Cela montre que la distribution conditionnelle n'a pas une grande variance en terme de scores de séparation. Ainsi prendre l'espérance comme estimation est un choix judicieux.

Tableau 4.4 – Tâche de séparation sur l'ensemble de test. Score SDR moyen (et écart-type) en dB et temps de calcul en ms pour la séparation de Voix et de Détonation pour les différentes méthodes : la méthode RASE (déclinée suivant les deux propositions DM-GMM et Def-MAP respectivement), la méthode RARE, ainsi que les méthodes concurrentes NMF et PLCA (non supervisée et supervisée respectivement) et le MSonyNet (réseau de neurones). L'objectif de temps-réel correspond à un temps de calcul maximum de 23ms. Plus le score SDR est élevé meilleur est le résultat.

Méthode		Param.	SDR Voix Min. 4,9 Max. 16,5	SDR Détonation Min. -4,6 Max. 10,4	Temps par frame (ms)
RASE	DM-GMM	10 comp.	8,9 (3,1)	2,3 (4,3)	5
		50 comp.	9,0 (3,3)	1,9 (4,7)	132
		100 comp.	9,4 (3,2)	2,6 (4,5)	470
RASE	Def-MAP	$\alpha = 10^{-4}$	1,8 (2,9)	-4,7 (4,9)	2553
		$\alpha = 10^{-3}$	2,6 (2,7)	-4,2 (4,7)	
		$\alpha = 10^{-2}$	3,2 (2,3)	-3,7 (5,0)	
		$\alpha = 10^{-1}$	3,4 (2,3)	-3,5 (5,0)	
		$\alpha = 1$	3,4 (2,3)	-3,5 (5,0)	
		$\alpha = 10$	3,4 (2,3)	-3,5 (5,0)	
RARE	-	-	4,5 (2,3)	-2,9 (5,0)	1
NMF	Non sup	2 comp.	3,7 (3,0)	-5,7 (4,7)	9
	Sup	100 comp.	8,3 (3,6)	3,1 (3,8)	9
PLCA	Non sup	2 comp.	3,2 (3,0)	-5,9 (4,5)	296
	Sup	100 comp.	8,3 (3,2)	1,6 (4,1)	296
MSonyNet	-	-	10,7 (3,7)	5,8 (5,1)	46

Tableau 4.5 – Étude de la variabilité de la distribution conditionnelle (4.40) de la proposition 1. Les scores sont les SDR, SIR et SAR moyens (et écart-type) en dB pour les deux sources d'intérêt Voix et Détonation.

SDR		SIR		SAR	
Voix	Détonation	Voix	Détonation	Voix	Détonation
3,82 (0,01)	-5,20 (0,22)	3,96 (0,01)	3,28 (0,54)	20,17 (0,04)	-2,86 (0,21)

4.5 Conclusion

Dans ce chapitre, nous avons présenté la méthode RASE qui permet de faire de la séparation audio en temps-réel, principalement dans le cas de deux jeux vidéos. Cette méthode est basée sur un modèle génératif des spectres complexes et deux méthodes de résolution : une par données manquantes et l'autre par déformation optimale des spectres complexes de la base d'apprentissage pour estimer les spectres observés.

Une contribution principale de cette partie est la proposition sur la déformation optimale de spectre, qui n'apparait pas dans l'état de l'art à l'heure actuelle. Cette proposition repose sur la déformation de couples de spectres complexes de l'ensemble d'apprentissage pour approcher au mieux le spectre mélange observé.

Les résultats préliminaires pour les deux propositions formulées plus haut suggèrent des perspectives de recherche intéressantes.

Concernant la proposition 1, l'usage des GMM pour les modèles de sources a été un moyen rapide et efficace pour le calcul des distributions conditionnelles. Néanmoins d'autres modélisations auraient pu être envisagées, comme notamment l'utilisation de lois non paramétriques afin de moins «contraindre» le modèle à suivre des hypothèses de normalités.

Concernant la proposition 2, la prochaine étape consiste à développer une méthode de réduction de données de manière à pouvoir réduire le temps de calcul et envisager de plus amples expérimentations. En effet, le principale avantage de la proposition 2, à savoir l'interprétabilité du modèle et des résultats, est un avantage non négligeable comparé à la proposition 1.

Chapitre 5

Conclusion générale et perspectives

5.1 Conclusion

Dans cette thèse nous avons proposé des méthodes probabilistes pour la classification et la séparation de sources sonores en temps-réel, dans le cadre d'un projet initié par un industriel. L'idée initiale était de proposer des améliorations du Sound Tracker d'A-Volute afin de le rendre plus intelligent, c'est-à-dire capable de localiser, séparer et classer les sources présentes dans une scène sonore complexe.

Le chapitre 3 présentait la méthode de classification développée dans ce but. Cette méthode permet de traiter les cas de classification monophonique (où une seule source est présente) et polyphonique (où plusieurs sources sont présentes en même temps) dans le cadre d'un modèle génératif. De plus, un attribut particulier, le spectre de puissance normalisé, a été considéré pour traiter à la fois les cas monophonique et polyphonique. Ce modèle génératif associé au spectre de puissance normalisé a permis de construire une règle de décision pour les deux cas en se basant uniquement sur la connaissance de sons monophoniques, ce qui est un avantage indéniable comparé aux autres méthodes concurrentes qui utilisent des bases de données de mélanges. De plus, toute la modélisation a été pensée pour le cas temps-réel, c'est-à-dire utiliser peu de données (durée de frame Δt et nombre de frames N petits) ainsi que peu gourmand en calculs (avec le prétraitement des noyaux). Les résultats expérimentaux notamment monophoniques, bien qu'encourageants, restent néanmoins perfectibles. En effet, un GMM bien réglé donne de meilleurs résultats que notre méthode sur le cas monophonique. Dans le cas de bases de données réduites notre méthode obtient des résultats intéressants, notamment en polyphonique comparé à des réseaux de neurones.

Le chapitre 4 présentait la méthode de séparation de sources sonores. Nous avons focalisé sur le cas de la séparation de deux sources de jeux vidéos (voix et détonation) dans un premier temps. Une première piste de recherche a été explorée consistant en une extension de la méthode de classification pour la séparation. Celle-ci se basait sur le paradigme des données manquantes et considérait les sources à retrouver comme les données manquantes sous condition que leur somme soit égale au mélange observé. Les premiers résultats montrent de bonnes performances comparées à d'autres méthodes de l'état de l'art. De plus le temps de calcul est modulable en choisissant un nombre adéquat de composantes pour le modèle. Une seconde piste de recherche consistait en l'utilisation de méthodes d'optimisation pour la recherche de déformation optimale

de spectres connus pour s'approcher au mieux du mélange observé. Son principal avantage est l'interprétabilité du modèle du fait des transformations appliquée, malgré les résultats mitigés sur l'ensemble de test.

5.2 Perspectives

Une flexibilité importante de notre méthode de classification étant le choix du noyau, il semble qu'il y ait encore des pistes de recherche à explorer pour le choix du noyau afin d'améliorer les résultats. En effet, le noyau multinomial, bien que flexible, semble amener du surapprentissage sur certaines bases de données mais également ne semble pas pertinent pour certaines classes (comme illustré sur la base ESC-10).

Concernant la séparation de sources, la proposition 1 utilisant les GMM, il serait intéressant de tester des lois non paramétriques ou d'autres lois de manière à modéliser plus finement les sources sonores considérées. En effet, le choix des GMM pour la proposition 1 était surtout un choix en terme de facilité de calculs, à la fois pour la résolution des équations mais également pour la mise en œuvre numérique. Concernant la proposition 2, une étape de réduction de données similaire à celle de la classification devrait être considérée pour pouvoir utiliser pleinement cette méthode. En effet, la mise en œuvre de la proposition 2 consiste à déformer des paires de spectres donc la complexité algorithmique est quadratique en le nombre de spectres à déformer. Une première tentative de réduction utilisant un algorithme de type k-means n'a pas montré de bons résultats dans ce cas. Il faudrait donc investiguer plus en détails cette perspective de recherche.

La partie sur la localisation n'a pas pu être traitée pendant ces trois années de thèse. Le sound Tracker de base réalise une localisation sur un flux multicanal en utilisant des indices acoustiques classiques comme les vecteurs de Gerzon [35], le B-format Ambisonic [54] et l'utilisation de l'intensité active et réactive acoustique [37]. Néanmoins des méthodes d'apprentissage statistiques ont également été developpées pour la localisation de sources sonores. L'étape de localisation peut également être utilisée comme a priori dans le cas de la séparation de sources multicanales car elle permet de focaliser dans certaines zones spatiales et d'extraire plus facilement les sources de la scène sonore.

Bibliographie

[1] S. Adavanne, P. Pertilä et T. Virtanen. « Sound event detection using spatial features and convolutional recurrent neural network ». In : *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, p. 771-775.

[2] S. Adavanne *et al.* « Sound event detection in multichannel audio using spatial and harmonic features ». In : *Detection and Classification of Acoustic Scenes and Events 2016*. Budapest, Hungary, 2016.

[3] H. Akaike. « A new look at the statistical model identification ». In : *IEEE Transactions on Automatic Control* 19.6 (déc. 1974), p. 716-723.

[4] R. Alazaidah, F. Thabtah et Q. Al-Radaideh. « A Multi-Label Classification Approach Based on Correlations Among Labels ». In : *International Journal of Advanced Computer Science and Applications* 6.2 (2015).

[5] M. Baelde, C. Biernacki et R. Greff. « A mixture model-based real-time audio sources classification method ». In : *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, p. 2427-2431.

[6] M. Baelde, C. Biernacki et R. Greff. « Classification de signaux audio en temps-réel par un modèle de mélanges d'histogrammes ». In : *49e Journées de Statistique*. Avignon, France, juin 2017.

[7] D. Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[8] T. Barker et T. Virtanen. « Blind Separation of Audio Mixtures Through Nonnegative Tensor Factorization of Modulation Spectrograms ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.12 (déc. 2016), p. 2377-2389.

[9] L. Benaroya, F. Bimbot et R. Gribonval. « Audio source separation with a single sensor ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (jan. 2006), p. 191-199.

[10] L. Benaroya et F. Bimbot. « Wiener based source separation with HMM/GMM using a single sensor ». In : *Proc. ICA*. 200. Citeseer. 2003, p. 957-96.

[11] E. Benetos *et al.* « Detection of overlapping acoustic events using a temporally-constrained probabilistic model ». In : *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2016, p. 6450-6454.

[12] C. Biernacki. *Mixture Models. Choix de modèles et agrégation, sous la direction de J-J. DROESBEKE, G. SAPORTA, C. THOMAS-AGNAN*. Technip. 2015.

[13] A. Bietti, F. Bach et A. Cont. « An online em algorithm in hidden (semi-)Markov models for audio segmentation and clustering ». In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2015, p. 1881-1885.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2006.

[15] V. Bisot, S. Essid et G. Richard. « Overlapping sound event detection with supervised Nonnegative Matrix Factorization ». In : *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, p. 31-35.

[16] D. Bogdanov *et al.* « Essentia : An audio analysis library for music information retrieval ». In : *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown] : ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR). 2013.

[17] L. Bottou. « Large-Scale Machine Learning with Stochastic Gradient Descent ». In : *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, 2010, p. 177-186.

[18] E. Cakir *et al.* « Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.6 (juin 2017), p. 1291-1303.

[19] E. Cakir, E. C. Ozan et T. Virtanen. « Filterbank learning for deep neural network based polyphonic sound event detection ». In : *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, p. 3399-3406.

[20] A. Camacho. « SWIPE : A SAWTOOTH WAVEFORM INSPIRED PITCH ESTIMATOR ». Thèse de doct. Université de Floride, 2007.

[21] C. Clavel, T. Ehrette et G. Richard. « Events Detection for an Audio-Based Surveillance System ». In : *2005 IEEE International Conference on Multimedia and Expo*. Juil. 2005, p. 1306-1309.

[22] K. Collins et P. J. Taillon. « Visualized sound effect icons for improved multimedia accessibility : A pilot study ». In : *Entertainment Computing* 3.1 (jan. 2012), p. 11-17.

[23] K. Crammer et Y. Singer. « On the algorithmic implementation of multiclass kernel-based vector machines ». In : *Journal of machine learning research* 2.Dec (2001), p. 265-292.

[24] A. P. Dempster, N. M. Laird et D. B. Rubin. « Maximum likelihood from incomplete data via the EM algorithm ». In : *Journal of the Royal Statistical Society, Series B* 39.1 (1977), p. 1-38.

[25] O. Dikmen et A. Mesaros. « Sound event detection using non-negative dictionaries learned from annotated overlapping events ». In : *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, p. 1-4.

[26] Z. Duan, G. J. Mysore et P. Smaragdis. « Online PLCA for real-time semi-supervised source separation ». In : *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2012, p. 34-41.

[27] N. Q. K. Duong, E. Vincent et R. Gribonval. « Spatial covariance models for under-determined reverberant audio source separation ». In : *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. Oct. 2009, p. 129-132.

[28] N. Duong, E. Vincent et R. Gribonval. *Under-determined reverberant audio source separation using a full-rank spatial covariance model*. Inria, 2010.

[29] T. T. H. Duong *et al.* « Gaussian Modeling-Based Multichannel Audio Source Separation Exploiting Generic Source Spectral Model ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.1 (jan. 2019), p. 32-43.

[30] T. T. H. Duong *et al.* « Multichannel audio source separation exploiting NMF-based generic source spectral model in Gaussian modeling framework ». In : *LVA-ICA*. 2 juil. 2018, p. 11.

[31] V. Emiya *et al.* « Subjective and Objective Quality Assessment of Audio Source Separation ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 19.7 (sept. 2011), p. 2046-2057.

[32] M. Espi *et al.* « Exploiting spectro-temporal locality in deep learning based acoustic event detection ». In : *EURASIP Journal on Audio, Speech, and Music Processing* 2015.1 (déc. 2015).

[33] C. Févotte, N. Bertin et J.-L. Durrieu. « Nonnegative Matrix Factorization with the Itakura-Saito Divergence : With Application to Music Analysis ». In : *Neural Computation* 21.3 (mar. 2009), p. 793-830.

[34] J. Flocon-Cholet. « Classification audio sous contrainte de faible latence ». Thèse de doct. Université de Rennes 1, 2016.

[35] M. M. Goodwin et J.-M. Jot. « A Frequency-domain Framework for Spatial Audio Coding Based on Universal Spatial Cues ». In : *AES Convention*. 1er mai 2006.

[36] E. M. Grais *et al.* « Two-Stage Single-Channel Audio Source Separation Using Deep Neural Networks ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.9 (sept. 2017), p. 1469-1479.

[37] R. N. Greff et H. C. T. Pham. « Method for visualizing the directional sound activity of a multichannel audio signal ». Brev. amér. 10085108B2. A-VOLUTE. 25 sept. 2018.

[38] Y. Han, J. Kim et K. Lee. « Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1 (jan. 2017), p. 208-221.

[39] T. Hastie, R. Tibshirani et J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY : Springer New York, 2009.

[40] T. Hayashi *et al.* « Duration-Controlled LSTM for Polyphonic Sound Event Detection ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.11 (nov. 2017), p. 2059-2070.

[41] T. Heittola *et al.* « Audio context recognition using audio event histograms ». In : *18th European Signal Processing Conference*. 23 août 2010, p. 1272-1276.

[42] T. Heittola *et al.* « Sound event detection in multisource environments using source separation ». In : *Workshop on machine listening in Multisource Environments*. 2011, p. 36-40.

[43] E. Hellinger. « Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen ». In : *J. Reine Angew. Math.* 136 (1909), p. 210-271.

[44] D. Istrate, M. Binet et S. Cheng. « Real time sound analysis for medical remote monitoring ». In : *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Août 2008, p. 4640-4643.

[45] C. Joder, S. Essid et G. Richard. « Temporal Integration for Audio Classification With Application to Musical Instrument Classification ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 17.1 (jan. 2009), p. 174-186.

[46] M. Kaminskas et F. Ricci. « Contextual music information retrieval and recommendation : State of the art and challenges ». In : *Computer Science Review* 6.2-3 (2012), p. 89-119.

[47] S. M. Kamruzzaman *et al.* « Speaker identification using mfcc-domain support vector machine ». In : *International Journal of Electrical and Power Engineering* 1 (2007), p. 274-278.

[48] S. M. Kay. *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[49] N. Keriven, A. Deleforge et A. Liutkus. « Blind Source Separation Using Mixtures of Alpha-Stable Distributions ». In : *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2018, p. 771-775.

[50] M. Kim et P. Smaragdis. « Collaborative audio enhancement using probabilistic latent component sharing ». In : *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, p. 896-900.

[51] D. P. Kingma et J. Ba. « Adam : A Method for Stochastic Optimization ». In : *ICLR*. 2015.

[52] D. Kounades-Bastian *et al.* « An inverse-gamma source variance prior with factorized parameterization for audio source separation ». In : *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, p. 136-140.

[53] G. Kour et N. Mehan. « Music Genre Classification using MFCC, SVM and BPNN ». In : *International Journal of Computer Applications* 112.6 (fév. 2015), p. 12-14.

[54] M.-V. Laitinen et V. Pulkki. « Binaural reproduction for Directional Audio Coding ». In : 2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). New Paltz, NY, USA : IEEE, oct. 2009, p. 337-340.

[55] S. Lecomte *et al.* « Abnormal events detection using unsupervised One-Class SVM - Application to audio surveillance and evaluation - ». In : *2011 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Août 2011, p. 124-129.

[56] Y. LeCun, Y. Bengio et G. Hinton. « Deep learning ». In : *Nature* 521.7553 (mai 2015), p. 436-444.

[57] D. Lee *et al.* « Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input ». In : *Detection and Classification of Acoustic Scenes and Events 2017*. 16 nov. 2017.

[58] H. Lee *et al.* « Unsupervised feature learning for audio classification using convolutional deep belief networks ». In : *Advances in Neural Information Processing Systems 22*. Sous la dir. d'Y. Bengio *et al.* 2009, p. 1096-1104.

[59] T.-W. Lee. « Independent Component Analysis ». In : *Independent Component Analysis*. Boston, MA : Springer, 1998, p. 27-66.

[60] D. Li *et al.* « Classification of general audio data for content-based retrieval ». In : *Pattern Recognition Letters*. Image/Video Indexing and Retrieval 22.5 (1er avr. 2001), p. 533-544.

[61] A. Liutkus *et al.* « The 2016 Signal Separation Evaluation Campaign ». In : *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*. Sous la dir. de P. Tichavský *et al.* Cham : Springer International Publishing, 2017, p. 323-332.

[62] Y. Luo et N. Mesgarani. « TaSNet : Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation ». In : *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2018, p. 696-700.

[63] Z. Ma et A. Leijon. « Modelling Speech Line Spectral Frequencies with Dirichlet Mixture Models ». In : *Interspeech 2010, 11th Annual Conference of the International Speech Communication Association*. Makuhari, Chiba, Japan, 26 sept. 2010.

[64] Z. Ma *et al.* « Bayesian estimation of Dirichlet mixture model with variational inference ». In : *Pattern Recognition* 47.9 (sept. 2014), p. 3143-3157.

[65] S. Makino, éd. *Audio Source Separation*. Signals and Communication Technology. Cham : Springer International Publishing, 2018.

[66] S. G. Mallat. « A theory for multiresolution signal decomposition : the wavelet representation ». In : *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (1989), p. 674-693.

[67] G. J. McLachlan et P. N. Jones. « Fitting Mixture Models to Grouped and Truncated Data via the EM Algorithm ». In : *Biometrics* 44.2 (1988), p. 571-578.

[68] A. Mesaros *et al.* « Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations ». In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2015, p. 151-155.

[69] A. Mesaros, T. Heittola et T. Virtanen. « Metrics for Polyphonic Sound Event Detection ». In : *Applied Sciences* 6.6 (25 mai 2016), p. 162.

[70] A. Mesaros, T. Heittola et T. Virtanen. « TUT database for acoustic scene classification and sound event detection ». In : *24th European Signal Processing Conference*. T. 2016. 2016.

[71] « Method and System for Visual Representation of Sound ». WO1999021169A1. K. Collins. 8 juil. 2010.

[72] « Method for visualizing the directional sound activity of a multichannel audio signal ». US2014/0177844A1. R. N. Greff et H. C. T. Pham. 26 juin 2014.

[73] S. Mirzaei, H. Van Hamme et Y. Norouzi. « Under-determined reverberant audio source separation using Bayesian Non-negative Matrix Factorization ». In : *Speech Communication*. Phase-Aware Signal Processing in Speech Communication 81 (juil. 2016), p. 129-137.

[74] Y. Mitsufuji, S. Koyama et H. Saruwatari. « Multichannel blind source separation based on non-negative tensor factorization in wavenumber domain ». In : *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, p. 56-60.

[75] G. Naithani *et al.* « Low-latency sound source separation using deep neural networks ». In : *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*. IEEE, 2016, p. 272-276.

[76] A. A. Nugraha, A. Liutkus et E. Vincent. « Multichannel Audio Source Separation With Deep Neural Networks ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.9 (sept. 2016), p. 1652-1664.

[77] A. E. Orhan et X. Pitkow. « Skip Connections Eliminate Singularities ». In : *arXiv preprint arXiv :1701.09175* (31 jan. 2017).

[78] A. Ozerov et C. Fevotte. « Multichannel Nonnegative Matrix Factorization in Convolutive Mixtures for Audio Source Separation ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 18.3 (mar. 2010), p. 550-563.

[79] A. Ozerov *et al.* « Adaptation of Bayesian Models for Single-Channel Source Separation and Its Application to Voice/Music Separation in Popular Songs ». In : *Trans. Audio, Speech and Lang. Proc.* 15.5 (juil. 2007), p. 1564-1578.

[80] A. Ozerov, E. Vincent et F. Bimbot. « A General Flexible Framework for the Handling of Prior Information in Audio Source Separation ». In : *IEEE Transactions on Audio, Speech and Language Processing* 20.4 (mai 2012), p. 1118-1133.

[81] D. Palaz, M. M. -Doss et R. Collobert. « Convolutional Neural Networks-based continuous speech recognition using raw speech signal ». In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2015, p. 4295-4299.

[82] S. J. Pan et Q. Yang. « A Survey on Transfer Learning ». In : *IEEE Transactions on Knowledge and Data Engineering* 22.10 (oct. 2010), p. 1345-1359.

[83] G. Parascandolo, H. Huttunen et T. Virtanen. « Recurrent neural networks for polyphonic sound event detection in real life recordings ». In : *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Mar. 2016, p. 6440-6444.

[84] G. Peeters *et al.* « The Timbre Toolbox : extracting audio descriptors from musical signals ». In : *The Journal of the Acoustical Society of America* 130.5 (nov. 2011), p. 2902-2916.

[85] K. J. Piczak. « Environmental sound classification with convolutional neural networks ». In : *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. Sept. 2015, p. 1-6.

[86] K. J. Piczak. « ESC : Dataset for Environmental Sound Classification ». In : ACM Press, 2015, p. 1015-1018.

[87] Y. Qian *et al.* « Very Deep Convolutional Neural Networks for Noise Robust Speech Recognition ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.12 (déc. 2016), p. 2263-2276.

[88] L. R. Rabiner. « A Tutorial on hidden Markov Models and Selected Applications in Speech Recognition ». In : *Proceedings of the IEEE* 77.2 (fév. 1989), p. 257-286.

[89] Z. Rafii *et al. The MUSDB18 corpus for music separation*. Déc. 2017.

[90] J. Read *et al.* « Classifier chains for multi-label classification ». In : *Machine Learning* 85.3 (30 juin 2011), p. 333.

[91] D. A. Reynolds. « An overview of automatic speaker recognition technology ». In : *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. T. 4. IEEE. 2002, p. IV-4072.

[92] D. E. Rumelhart, G. E. Hinton et R. J. Williams. « Learning representations by back-propagating errors ». In : *Nature* 3.23 (9 oct. 1986), p. 533-536.

[93] P. Ruvolo, I. Fasel et J. R. Movellan. « A learning approach to hierarchical feature selection and aggregation for audio classification ». In : *Pattern Recognition Letters*. Pattern Recognition of Non-Speech Audio 31.12 (1er sept. 2010), p. 1535-1542.

[94] J. Salamon et J. P. Bello. « Unsupervised feature learning for urban sound classification ». In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2015, p. 171-175.

[95] H. Sawada, S. Araki et S. Makino. « Underdetermined Convolutive Blind Source Separation via Frequency Bin-Wise Clustering and Permutation Alignment ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 19.3 (mar. 2011), p. 516-527.

[96] N. Scaringella et D. Mlynek. « A mixture of support vector machines for audio classification ». In : *IEEE MIREX, London* (2005).

[97] G. Schwarz. « Estimating the Dimension of a Model ». In : *The Annals of Statistics* 6.2 (mar. 1978), p. 461-464.

[98] C. E. Shannon. « Communication in the presence of noise ». In : *Proceedings of the Institute of the Radio Engineers* 37.1 (jan. 1949), p. 10-21.

[99] M. Shashanka. « Latent Variable Framework for Modeling and Separating Single-Channel Acoustic Sources ». Thèse de doct. Boston University, 2008.

[100] L. S. R. Simon et E. Vincent. *Combining blockwise and multi-coefficient stepwise approches in a general framework for online audio source separation*. RR8766. Inria, 2015, p. 18.

[101] U. Simsekli, A. Liutkus et A. T. Cemgil. « Alpha-Stable Matrix Factorization ». In : *IEEE Signal Processing Letters* 22.12 (déc. 2015), p. 2289-2293.

[102] J. O. Smith III. *Spectral Audio Signal Processing*. W3K Publishing, 2011.

[103] J. O. Smith III. *Physical Audio Signal Processing*. W3K Publishing, 2010.

[104] G. Soromenho. « Comparing approaches for testing the number of components in a finite mixture model ». In : *Computational Statistics* 9.1 (1er jan. 1994).

[105] N. Srivastava *et al.* « Dropout : A Simple Way to Prevent Neural Networks from Overfitting ». In : *Journal of Machine Learning Research* 15 (2014), p. 1929-1958.

[106] D. Stoller, S. Ewert et S. Dixon. « Adversarial Semi-Supervised Audio Source Separation Applied to Singing Voice Extraction ». In : *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2018, p. 2391-2395.

[107] F.-R. Stöter, A. Liutkus et N. Ito. « The 2018 Signal Separation Evaluation Campaign ». In : *LVA ICA : Latent Variable Analysis and Signal Separation*. 14th International Conference on Latent Variable Analysis and Signal Separation. Surrey, United Kingdom, juil. 2018, p. 13.

[108] N. Takahashi et Y. Mitsufuji. « Multi-Scale multi-band densenets for audio source separation ». In : *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. Oct. 2017, p. 21-25.

[109] N. Takahashi, N. Goswami et Y. Mitsufuji. « MMDenseLSTM : An efficient combination of convolutional and recurrent neural networks for audio source separation ». In : *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. Sept. 2018.

[110] N. Takahashi *et al.* « Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Detection ». In : *Interspeech 2016*. Sept. 2016.

[111] J. Thiemann, N. Ito et E. Vincent. *DEMAND : a collection of multi-channel recordings of acoustic noise in diverse environments*. 2018.

[112] V. Tiwari. « MFCC and its applications in speaker recognition ». In : *International journal on emerging technologies* 1.1 (2010), p. 19-22.

[113] G. Tsoumakas et I. Vlahavas. « Random k-Labelsets : An Ensemble Method for Multilabel Classification ». In : *Machine Learning : ECML 2007* 4701 (2007), p. 406-417.

[114] S. Uhlich *et al.* « Improving music source separation based on deep neural networks through data augmentation and network blending ». In : *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, p. 261-265.

[115] S. Uhlich, F. Giron et Y. Mitsufuji. « Deep neural network based instrument extraction from music ». In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Avr. 2015, p. 2135-2139.

[116] A. Van Der Oord, S. Dieleman et B. Schrauwen. « Transfer learning by supervised pre-training for audio-based music classification ». In : 15th International Society for Music Information Retrieval Conference. 2014, p. 29-34.

[117] M. Vetterli, J. Kovačević et V. K. Goyal. *Foundations of signal processing*. Cambridge : Cambridge University Press, 2014. 715 p.

[118] E. Vincent, R. Gribonval et C. Fevotte. « Performance measurement in blind audio source separation ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 14.4 (juil. 2006), p. 1462-1469.

[119] E. Vincent, S. Arberet et R. Gribonval. « Underdetermined instantaneous audio source separation via local Gaussian modeling ». In : *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2009, p. 775-782.

[120] Z. Wang et X. Xue. « Multi-Class Support Vector Machine ». In : *Support Vector Machines Applications*. Nov. 2013, p. 23-48.

[121] J. H. J. Ward. « Hierarchical Grouping to Optimize and Objective Function ». In : *Journal of the American Statistical Association* 58.301 (mar. 1963), p. 236-244.

[122] J. Weston, C. Watkins *et al.* « Support vector machines for multi-class pattern recognition. » In : *Esann*. T. 99. 1999, p. 219-224.

[123] Y. Xu *et al.* « Unsupervised feature learning based on deep models for environmental audio tagging ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2017).

[124] K. Yoshii *et al.* « Infinite Positive Semidefinite Tensor Factorization for Source Separation of Mixture Signals. » In : *ICML (3)*. 2013, p. 576-584.

[125] N. Zhang, J. Yan et Y. Zhou. « Unsupervised Audio Source Separation via Spectrum Energy Preserved Wasserstein Learning ». In : *CoRR* (11 nov. 2017).

[126] P. Zhang, X. Ma et S. Ding. « Audio Source Separation from a Monaural Mixture Using Convolutional Neural Network in the Time Domain ». In : *Advances in Neural Networks - ISNN 2017*. Lecture Notes in Computer Science (2017).

[127] X. Zhang et D. Wang. « Deep Learning Based Binaural Speech Separation in Reverberant Environments ». In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2017), p. 1-1.

Annexe A

Liste des descripteurs audio

La classification audio utilise un ensemble d'attributs appelés *descripteurs audio* [84]. Ces descripteurs sont extraits de différentes représentations de signaux : temporelle, spectrale, harmonique, cepstrale et perceptuelle.

A.1 Descripteurs temporels

Les descripteurs temporels se calculent à partir d'une représentation temporelle du signal $\mathbf{x} \in \mathbb{R}^N$ et son enveloppe d'énergie $\mathbf{e} = |\mathbf{x}|^{\cdot 2}$, où $|\cdot|$ et $\cdot^2$ sont le module et la mise au carré élément par élément, et $\mathbf{t} = (t_n)$ le vecteur des temps discrétisés.

Les coefficients d'*autocorrélation* sont calculés avec la formule suivante :

$$\gamma_c = \frac{1}{\gamma_0} \sum_{n=0}^{N-c-1} x_n x_{n+c}. \tag{A.1}$$

Ceux-ci peuvent être interprétés comme la distribution spectrale du signal dans le domaine temporel (propriété de la transformée de Fourier).

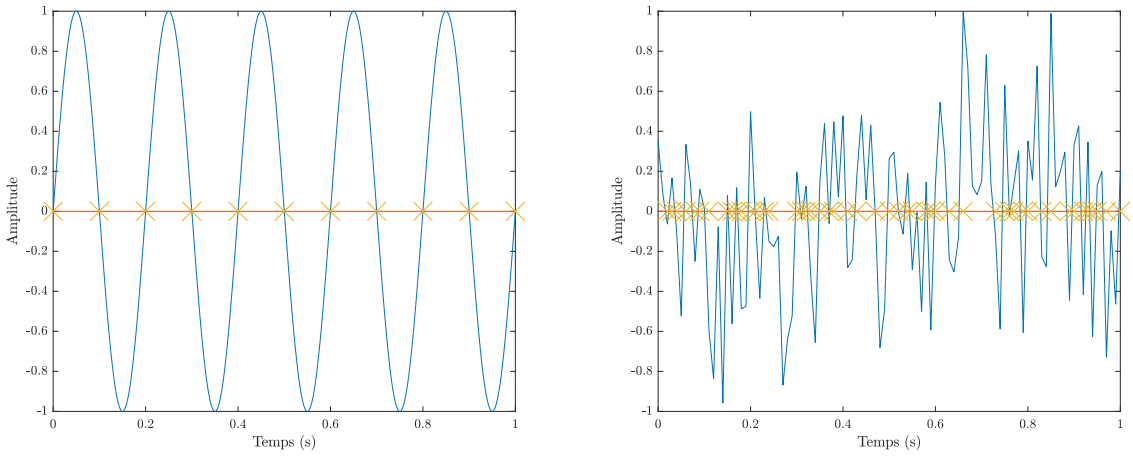
Le *taux de passage par zéro* (ZCR) est le nombre de fois que le signal passe par zéro. Il est utile pour distinguer des bruits (fort ZCR) d'un son périodique (faible ZCR) (voir Figure A.1).

Plusieurs descripteurs sont liés à l'attaque d'un son (c'est-à-dire le fait qu'un son passe du silence à son amplitude maximum). Le *temps d'attaque* est le temps mis par le signal pour atteindre son premier maximum depuis le temps initial. Le *log-temps d'attaque* est plus souvent considéré au lieu du temps d'attaque. La *pente de l'attaque* est la pente moyennée temporellement de l'enveloppe d'énergie. La *pente de décroissance* et le taux avec lequel le signal décroît. On considère un modèle exponentiel pour l'enveloppe d'énergie et cette pente est modélisée par le coefficient α :

$$e_n = A e^{-\alpha(t_n - t_N)}. \tag{A.2}$$

Le *centroide temporel* est le barycentre de l'enveloppe d'énergie :

$$\mathrm{TC} = \frac{\sum_{n=0}^{N-1} t_n e_n}{\sum_{n=0}^{N-1} t_n}. \tag{A.3}$$



(a) (b)

FIGURE A.1 – Illustration du ZCR sur deux signaux différents. (a) Pour un signal sinusoidal le ZCR est plus tôt faible. (b) Pour un signal bruité le ZCR est plutôt fort.

La *modulation d'énergie* est la modulation de la partie type sustain.

A.2 Descripteurs spectraux

Les descripteurs spectraux se calculent à partir d'une représentation spectrale d'un signal, en particulier l'amplitude du spectre $\mathbf{a} \in \mathbb{R}^N$ et l'amplitude normalisée $\mathbf{p} = \mathbf{a}./\sum_{k=0}^{N-1} a_k$, où ./ est la division élément par élement, et $\mathbf{f} = (f_k)$ le vecteur de fréquences discrétisées.

L'*énergie totale* est définie par :

$$E_T = \sum_{k=0}^{N-1} a_k^2. \tag{A.4}$$

Les *moments statistiques* sont le centroide μ_1, le spread μ_2, le skewness μ_3 et le kurtosis μ_4 de l'amplitude normalisée sur spectre :

$$\mu_1 = \sum_{k=0}^{N-1} f_k p_k, \tag{A.5}$$

$$\mu_2 = \sum_{k=0}^{N-1} \left((f_k - \mu_1)^2 p_k \right)^{1/2}, \tag{A.6}$$

$$\mu_3 = \sum_{k=0}^{N-1} \left((f_k - \mu_1)^3 p_k \right)^{1/2} \Big/ \mu_2^3, \tag{A.7}$$

$$\mu_4 = \sum_{k=0}^{N-1} \left((f_k - \mu_1)^4 p_k \right)^{1/2} \Big/ \mu_2^4. \tag{A.8}$$

Les descripteurs suivants décrivent la forme du spectre. La *pente spectrale* est le coefficient d'une régression linéaire sur l'amplitude du spectre. La *décroissance spectrale* est la moyenne des pentes spectrales. Le *roll-off spectral* est la fréquence à laquelle 95% de l'énergie est atteinte. Le *flatness spectral* est le ratio entre la moyenne géométrique et la moyenne arithmétique du spectre :

$$\mathrm{SF} = \frac{\left(\prod_{k=0}^{N-1} a_k \right)^{1/N}}{\frac{1}{N} \sum_{n=0}^{N-1} a_k}. \tag{A.9}$$

Il permet de quantifier le fait qu'un spectre est «plat». Le *crest spectral* est le ratio entre la valeur maximum et la moyenne arithmétique du spectre :

$$\mathrm{SC} = \frac{\max_k a_k}{\frac{1}{N} \sum_{n=0}^{N-1} a_k}. \tag{A.10}$$

Il permet de quantifier le fait qu'un spectre présente des «pics».

A.2.1 Descripteurs harmoniques

Les descripteurs harmoniques se calculent à partir d'une représentation harmonique du signal, c'est-à-dire de la forme :

$$x_n = \sum_{h=1}^{H} a_h cos(2\pi f_h t_n + \phi_h), \tag{A.11}$$

avec H le nombre d'harmoniques, $a_n(h)$ les amplitudes, f_h les fréquences harmoniques et ϕ_h les déphasages.

La *fréquence fondamentale* est la première fréquence de la décomposition harmonique f_0. La fréquence fondamentale peut être estimée avec la technique Swipe de Camacho [20].

L'*énergie harmonique* est l'énergie totale des harmoniques :

$$E_H = \sum_{h=1}^{H} a_h^2. \tag{A.12}$$

L'*énergie du bruit* est l'énergie résiduelle qui n'est pas expliquée par la décomposition harmonique :

$$E_N = E_T - E_H. \tag{A.13}$$

Le *noisiness* est le ratio entre l'énergie du bruit et l'énergie totale :

$$\text{noisiness} = \frac{E_N}{E_T}. \tag{A.14}$$

Le *tristimulus* est l'équivalent timbral des couleurs représenté par trois quantités :

$$\begin{aligned} T_1 &= \frac{a_1}{\sum_{h=1}^{H} a_h}, \\ T_2 &= \frac{\sum_{h=1}^{3} a_h}{\sum_{h=1}^{H} a_h}, \\ T_3 &= \frac{\sum_{h=4}^{H} a_h}{\sum_{h=1}^{H} a_h}. \end{aligned} \tag{A.15}$$

L'*inharmonicité* est la différence entre les fréquences harmoniques présentes f_h et les harmoniques «pures» hf_0 :

$$\text{inharmo} = \frac{2}{f_0} \frac{\sum_{h=1}^{H} (f_h - hf_0) a_h^2}{\sum_{h=1}^{H} a_h^2}. \tag{A.16}$$

La *déviation spectrale harmonique* HDEV est définie comme la déviation entre le modèle harmonique et l'enveloppe spectrale SE :

$$\begin{aligned} \text{HDEV} &= \frac{1}{H} \sum_{h=1}^{H} (a_h - SE(f_h)), \\ \text{SE}(f_h) &= \frac{1}{3}(a_{h-1} + a_h + a_{h+1}). \end{aligned} \tag{A.17}$$

Le *ratio d'énergie harmonique impaire-à-paire* est le ratio entre les harmoniques impaires et les harmoniques paires :

$$\text{OER} = \frac{\sum_{h=1}^{H/2} a_{2h-1}^2}{\sum_{h=1}^{H/2} a_{2h}^2}. \tag{A.18}$$

La *variation spectrale* est définie comme 1 moins la corrélation entre deux frames successives.

A.3 Descripteurs cepstraux & perceptuels

Les descripteurs cepstraux se calculent à partir de la représentation cepstrale d'un signal. Le cepstre est défini comme la TCD (Transformée en cosinus discrète) de la log-amplitude de la TFD d'un signal :

$$\mathbf{C_x} = f_{\text{TCD}}\left(\log_{10} |f_{\text{TFD}}(\mathbf{x})|\right). \tag{A.19}$$

Les descripteurs perceptuels se calculent à partir de différentes échelles fréquentielles. L'échelle par défaut est l'échelle *linéaire* ou échelle Hertz. Cependant, l'audition humaine est plus sensible à une échelle *logarithmique*, et trois échelles spécifiques sont définies :

— L'échelle ERB est une approximation rectangulaire d'une banque de filtres de la forme :

$$ERB(f) = 6,23f^2 + 93,39f + 28,52,$$

— L'échelle Mel est linéaire pour $f < 1$kHz et logarithme pour $f > 1$kHz :

$$M(f) = 1000\left(1 + \log_{10}\left(\frac{f}{1000}\right)\right),$$

— L'échelle Bark est une approximation plus précise de l'audition humaine :

$$B(f) = 13\arctan\left(\frac{f}{1315,8}\right) + 3,5\arctan\left(\frac{f}{7518}\right).$$

Basés sur ces échelles perceptuelles, on peut définir les MFCC (Mel Frequency Cepstral Coefficients), BFCC (Bark Frequency Cepstral Coefficients) et ERBFCC (ERB Frequency Cepstral Coefficients). Les MFCC sont souvent utilisés en classification et traitement audio car ils résument le spectre avec un ensemble très réduit de coefficients (typiquement 12 coefficients sont utilisés).

Annexe B

Détails de certains calculs

Cette annexe présente des détails techniques de calculs concernant des estimations ou des développements de certaines propriétés développés dans la thèse. Il y est notamment présenté la dérivation des paramètres d'un GMM estimé avec des données par intervalle ou encore l'estimation d'un mélange de Dirichlet.

B.1 Dérivation des paramètres d'un GMM dans le cas par intervalle

Ces calculs sont également disponibles dans [67]. On rappelle les notations données au Chapitre 2 concernant les GMM par intervalle. L'espace d'échantillonnage $\mathcal{X}$ de $\mathbf{X}$ est divisé en r sous-ensembles mutuellement exclusifs $\mathcal{X}_j$. De plus, on ne connait que le nombre n_j d'observations de $\mathbf{X}$ tombant dans chaque $\mathcal{X}_j$. On note $n = \sum_{j=1}^{r} n_j$, et on suppose que $\mathbf{y} = (n_1, ..., n_r)$ suit une loi multinomiale de n tirages sur r catégories, de probabilités $P_j(\boldsymbol{\theta})/P(\boldsymbol{\theta})$, avec :

$$\begin{aligned} P_j(\boldsymbol{\theta}) &= \int_{\mathcal{X}_j} p(\mathbf{x}; \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}, \\ P(\boldsymbol{\theta}) &= \sum_{j=1}^{r} P_j(\boldsymbol{\theta}). \end{aligned} \tag{B.1}$$

La log-vraisemblance s'écrit dans ce cas :

$$\mathcal{L}(\boldsymbol{\theta}) = \left(\frac{n!}{\prod_{j=1}^{r} n_j!} \right) \prod_{j=1}^{r} \left[\frac{P_j(\boldsymbol{\theta})}{P(\boldsymbol{\theta})} \right]^{n_j}. \tag{B.2}$$

Pour se mettre dans le cadre de l'algorithme EM, on introduit les variables $\mathbf{x}_j$ qui correspondent aux données observées, $\mathbf{w}$ qui correspond aux données observées et leurs comptages ainsi que $\mathbf{z}_{jg}$ les labels représentant l'appartenance de la variable x_{jg} au groupe k :

$$\mathbf{x}_j = \left(x_{j1},...,x_{jn_j}\right),\ j=1,...,r, \tag{B.3}$$
$$\mathbf{w} = (\mathbf{y},\mathbf{x}_1,...,\mathbf{x}_r), \tag{B.4}$$
$$\mathbf{z}_{jg} = \left(z_{1jg},...,z_{Kjg}\right),\ j=1,...,r; g=1,...,n_j, \tag{B.5}$$

avec la contrainte $\sum_{k=1}^{K} z_{kjg} = 1$. On a, de même que pour un EM dans le cas GMM classique, les probabilités d'appartenance aux groupes suivantes :

$$p\left(z_{kjg}=1|x_{jg}\right) = \frac{\pi_k p_k\left(x_{jg};\boldsymbol{\theta}\right)}{p\left(x_{jg};\boldsymbol{\theta}\right)} = \tau_k\left(x_{jg};\boldsymbol{\theta}\right). \tag{B.6}$$

L'algorithme EM va donc chercher à maximiser la log-vraisemblance des données complètes suivante :

$$\ell_c(\boldsymbol{\theta}) = \sum_{k=1}^{K}\sum_{j=1}^{r}\sum_{g=1}^{n_j} z_{kjg}\left(\log p_k\left(x_{jg};\boldsymbol{\theta}\right) + \log\pi_k\right). \tag{B.7}$$

L'étape E consiste à calculer :

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{k=1}^{K}\sum_{j=1}^{r} n_j \mathbb{E}_j^{(t)}\left[\tau_k(\mathbf{X};\boldsymbol{\theta})\left(\log p_k(\mathbf{X};\boldsymbol{\theta}) + \log\pi_k\right)\right]. \tag{B.8}$$

L'espérance précédente ne sera calculée qu'au cas par cas lorsqu'un domaine de données particulier sera donné. En effet dans le cas général cette espérance n'a pas de formule close. La mise à jour des paramètres à l'étape M est :

$$\pi_k^{(t+1)} = \frac{\sum_{j=1}^{r} n_j \mathbb{E}_j^{(t)}[\tau_k(\mathbf{X};\boldsymbol{\theta})]}{\sum_{j=1}^{r} n_j}, \tag{B.9}$$
$$\mu_k^{(t+1)} = \frac{\sum_{j=1}^{r} n_j \mathbb{E}_j^{(t)}[\tau_k(\mathbf{X};\boldsymbol{\theta})\mathbf{X}]}{\sum_{j=1}^{r} n_j \mathbb{E}_j^{(t)}[\tau_k(\mathbf{X};\boldsymbol{\theta})]}, \tag{B.10}$$
$$\left[\sigma_k^{(t+1)}\right]^2 = \frac{\sum_{j=1}^{r} n_j \mathbb{E}_j^{(t)}\left[\tau_k(\mathbf{X};\boldsymbol{\theta})\left(\mathbf{X}-\mu_k^{(t+1)}\right)^2\right]}{\sum_{j=1}^{r} n_j \mathbb{E}_j^{(t)}[\tau_k(\mathbf{X};\boldsymbol{\theta})]}. \tag{B.11}$$

Dans le cas où les $\mathcal{X}_j$ sont des intervalles (a_j,b_j), les expérances $A_{s,k,j}^{(t)} = \mathbb{E}_j^{(t)}[\tau_k(X|\Theta)X^s]$ $(s=0,1)$ et $A_{2,k,j}^{(t)} = \mathbb{E}_j^{(t)}\left[\tau_k(X|\Theta)\left(X-\mu_k^{(t+1)}\right)^2\right]$ s'écrivent :

$$A_{s,g,j}^{(t)} = \frac{\pi_g^{(t)} G_{s,g,j}^{(t)}}{F^{(t)}(b_j) - F^{(t)}(a_j)}, \tag{B.12}$$

avec les variables $G_{s,g,j}^{(t)}$ définies par :

$$G^{(t)}_{0,g,j} = H_{0,g,j}, \tag{B.13}$$

$$G^{(t)}_{1,g,j} = \mu^{(t)}_g H^{(t)}_{0,g,j} - \left(\sigma^{(t)}_g\right)^2 H^{(t)}_{1,g,j}, \tag{B.14}$$

$$G^{(t)}_{2,g,j} = \left(\sigma^{(t)}_g\right)^2 \left(H^{(t)}_{0,g,j} + \left(2\mu^{(t+1)}_g - \mu^{(t)}_g\right) H^{(t)}_{1,g,j} - H^{(t)}_{2,g,j}\right) + \left(\mu^{(t+1)}_g - \mu^{(t)}_g\right)^2 H_{0,g,j}, \tag{B.15}$$

et les variables $H^{(t)}_{s,g,j}$ définies par :

$$H^{(t)}_{0,g,j} = F^{(t)}_g(b_j) - F^{(t)}_g(a_j), \tag{B.16}$$

$$H^{(t)}_{1,g,j} = p^{(t)}_g(b_j) - p^{(t)}_g(a_j), \tag{B.17}$$

$$H^{(t)}_{2,g,j} = b_j p^{(t)}_g(b_j) - a_j p^{(t)}_g(a_j). \tag{B.18}$$

On note également les densités de probabilité ainsi que les fonctions de répartition de la manière suivante :

$$p^{(t)}_g(x) = \left((2\pi)^{1/2}\sigma^{(t)}_g\right)^{-1} \exp\left(-\frac{1}{2}\left((x-\mu^{(t)}_g)/\sigma^{(t)}_g\right)^2\right), \tag{B.19}$$

$$F^{(t)}_g(x) = \int_{-\infty}^{x} p^{(t)}_g(x)\mathrm{d}x, \tag{B.20}$$

$$F^{(t)}(x) = \sum_{g=1}^{G} \pi^{(t)}_g F^{(t)}_g(x). \tag{B.21}$$

B.2 Détails sur les mélanges de Dirichlet

On rappelle l'expression du mélange de Dirichlet :

$$p(\mathbf{x};\boldsymbol{\theta}) = \sum_{g=1}^{G} \pi_g \mathrm{Dir}\left(\mathbf{x};\boldsymbol{\alpha}_g\right), \tag{B.22}$$

avec $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_G\}$. Une première manière d'estimer ce mélange est de passer par un algorithme EM classique, comme dans [63]. L'étape E consiste à calculer les espérances des variables latentes $\overline{z}_{ig} = \mathbb{E}\left(z_{ig}\right)$:

$$\overline{z}_{ig} = \frac{\pi_g \mathrm{Dir}\left(\mathbf{x}_i;\boldsymbol{\alpha}_g\right)}{\sum_{g'=1}^{G} \pi_{g'} \mathrm{Dir}\left(\mathbf{x}_i;\boldsymbol{\alpha}_{g'}\right)}. \tag{B.23}$$

Ensuite pour l'étape M, les proportions sont facilement estimées :

$$\widehat{\pi}_g = \frac{1}{n}\sum_{i=1}^{n} \overline{z}_{ig}, \tag{B.24}$$

En revanche l'estimation des paramètres des lois de Dirichlet se fait en résolvant le système

d'équations suivant :

$$\begin{bmatrix} \psi\left(\alpha_{1g}\right) - \psi\left(\sum_{b=1}^{B} \alpha_{bg}\right) \\ \vdots \\ \psi\left(\alpha_{Bg}\right) - \psi\left(\sum_{g=1}^{G} \alpha_{bg}\right) \end{bmatrix} = \begin{bmatrix} \frac{\sum_{i=1}^{n} \overline{z}_{ig} \log x_{1i}}{\sum_{i=1}^{n} \overline{z}_{ig}} \\ \vdots \\ \frac{\sum_{i=1}^{n} \overline{z}_{ig} \log x_{Bi}}{\sum_{i=1}^{n} \overline{z}_{ig}} \end{bmatrix}. \tag{B.25}$$

Ce système peut être résolu avec la méthode de Newton-Raphson par exemple. En pratique c'est cette méthode que nous avons utilisée pour les expériences lors de la thèse.

Une méthode alternative utilisant l'inférence variationnelle a également été considérée dans [64]. Néanmoins l'apprentissage d'un DMM avec cette méthode était trop long comparé à la méthode précédente et n'a pas été retenu.

B.3 Séparation : calcul des distributions hiérarchiques

Commençons par la première distribution. Sous l'hypothèse gaussienne, on a :

$$p_{12}\left(\overline{\mathbf{x}}_{12}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \pi_{1g_1} \pi_{2g_2} \mathcal{N}\left(\overline{\mathbf{x}}_{12}^{\text{freq}}; \boldsymbol{\mu}_{1g_1} + \boldsymbol{\mu}_{2g_2}, \Sigma_{1g_1} + \Sigma_{2g_2}\right), \tag{B.26}$$

$$p_3\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_{12}^{\text{freq}}\right) = \sum_{g_3=1}^{G_3} \pi_{3g_3} \mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}} - \overline{\mathbf{x}}_{12}^{\text{freq}}; \boldsymbol{\mu}_{3g_3}, \Sigma_{3g_3}\right), \tag{B.27}$$

$$p\left(\overline{\mathbf{x}}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \sum_{g_3=1}^{G_3} \left(\prod_{i=1}^{3} \pi_{ig_i}\right) \mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}; \sum_{i=1}^{3} \boldsymbol{\mu}_{ig_i}, \sum_{i=1}^{3} \Sigma_{ig_i}\right). \tag{B.28}$$

La première distribution conditionnelle donne donc :

$$p\left(\overline{\mathbf{x}}_{12}^{\text{freq}} | \overline{\mathbf{x}}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1} \sum_{g_2=1}^{G_2} \sum_{g_3=1}^{G_3} \varphi_{g_1 g_2 g_3}\left(\overline{\mathbf{x}}^{\text{freq}}\right) \mathcal{N}\left(\overline{\mathbf{x}}_{12}^{\text{freq}}; \tilde{\boldsymbol{\mu}}_{g_1 g_2 g_3}, \tilde{\Sigma}_{g_1 g_2 g_3}\right), \tag{B.29}$$

avec les quantités suivantes :

$$\varphi_{g_1 g_2 g_3}\left(\overline{\mathbf{x}}^{\text{freq}}\right) = \frac{\left(\prod_{i=1}^{3} \pi_{ig_i}\right) \mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}; \sum_{i=1}^{3} \boldsymbol{\mu}_{ig_i}, \sum_{i=1}^{3} \Sigma_{ig_i}\right)}{\sum_{g_1'=1}^{G_1} \sum_{g_2'=1}^{G_2} \sum_{g_3'=1}^{G_3} \left(\prod_{i=1}^{3} \pi_{ig_i'}\right) \mathcal{N}\left(\overline{\mathbf{x}}^{\text{freq}}; \sum_{i=1}^{3} \boldsymbol{\mu}_{ig_i'}, \sum_{i=1}^{3} \Sigma_{ig_i'}\right)}, \tag{B.30}$$

$$\tilde{\boldsymbol{\mu}}_{g_1 g_2 g_3} = \sum_{i=1}^{2} \boldsymbol{\mu}_{ig_i} + \left(\sum_{i=1}^{2} \Sigma_{ig_i}\right)\left(\sum_{i=1}^{3} \Sigma_{ig_i}\right)^{-1} \left(\overline{\mathbf{x}}^{\text{freq}} - \sum_{i=1}^{3} \boldsymbol{\mu}_{ig_i}\right), \tag{B.31}$$

$$\tilde{\Sigma}_{g_1 g_2 g_3} = \sum_{i=1}^{2} \Sigma_{ig_i} - \left(\sum_{i=1}^{2} \Sigma_{ig_i}\right)\left(\sum_{i=1}^{3} \Sigma_{ig_i}\right)^{-1} \left(\sum_{i=1}^{2} \Sigma_{ig_i}\right). \tag{B.32}$$

Ensuite, on utilise les résultats de la séparation à deux sources pour obtenir la deuxième distribution conditionnelle :

$$p\left(\overline{\mathbf{x}}_1^{\text{freq}}|\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}\right) = \sum_{g_1=1}^{G_1}\sum_{g_2=1}^{G_2} \varphi_{g_1g_2}\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}\right)\mathcal{N}\left(\overline{\mathbf{x}}_1^{\text{freq}}; \tilde{\boldsymbol{\mu}}_{g_1g_2}, \tilde{\Sigma}_{g_1g_2}\right) \tag{B.33}$$

avec les quantités suivantes :

$$\varphi_{g_1g_2}\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}\right) = \frac{\pi_{1g_1}\pi_{2g_2}\mathcal{N}\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}; \boldsymbol{\mu}_{1g_1}+\boldsymbol{\mu}_{2g_2}, \Sigma_{1g_1}+\Sigma_{2g_2}\right)}{\sum_{g_1'=1}^{G_1}\sum_{g_2'=1}^{G_2}\pi_{1g_1'}\pi_{2g_2'}\mathcal{N}\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}}; \boldsymbol{\mu}_{1g_1'}+\boldsymbol{\mu}_{2g_2'}, \Sigma_{1g_1'}+\Sigma_{2g_2'}\right)}, \tag{B.34}$$

$$\tilde{\boldsymbol{\mu}}_{g_1g_2} = \boldsymbol{\mu}_{1g_1} + \Sigma_{1g_1}\left(\Sigma_{1g_1}+\Sigma_{2g_2}\right)^{-1}\left(\widehat{\overline{\mathbf{x}}}_{12}^{\text{freq}} - \boldsymbol{\mu}_{1g_1} - \boldsymbol{\mu}_{2g_2}\right), \tag{B.35}$$

$$\tilde{\Sigma}_{g_1g_2} = \Sigma_{1g_1} - \Sigma_{1g_1}\left(\Sigma_{1g_1}+\Sigma_{2g_2}\right)^{-1}\Sigma_{1g_1}. \tag{B.36}$$

B.4 Séparation : optimisation de déformation en amplitude

On rappelle le problème d'optimisation à résoudre :

$$\min_{\mathbf{T}_1,\mathbf{T}_2\in\mathbf{R}^B} \|\mathbf{T}_1-\mathbf{1}\|^2 + \|\mathbf{T}_2-\mathbf{1}\|^2 \tag{B.37}$$

$$\text{s.c. } \mathbf{T}_1\cdot|\mathbf{x}_1^{\text{freq}}| + \mathbf{T}_2\cdot|\mathbf{x}_2^{\text{freq}}| = |\mathbf{x}^{\text{freq}}|. \tag{B.38}$$

On passe par la formulation des multiplicateurs de Lagrange :

$$\min_{\mathbf{T}_1,\mathbf{T}_2\in\mathbf{R}^B} \mathbb{L}(\mathbf{T}_1,\mathbf{T}_2,\boldsymbol{\lambda}) = \|\mathbf{T}_1-\mathbf{1}\|^2 + \|\mathbf{T}_2-\mathbf{1}\|^2 + \sum_{b=1}^{B}\lambda_b\left(T_{1b}|x_{1b}^{\text{freq}}| + T_{2b}|x_{2b}^{\text{freq}}| - |x_b^{\text{freq}}|\right). \tag{B.39}$$

On dérive par rapport à chaque variable :

$$\frac{\partial\mathbb{L}}{\partial T_{zb}} = 2T_{zb} - 2 + \lambda_b|x_{zb}^{\text{freq}}|. \tag{B.40}$$

On résout par rapport à chaque variable en annulant le gradient :

$$T_{zb} = 1 - \frac{1}{2}\lambda_b|x_{zb}^{\text{freq}}|. \tag{B.41}$$

On utilise la contrainte pour trouver le paramètre de Lagrange :

$$x_b = \left(1-\frac{1}{2}\lambda_b|x_{1b}^{\text{freq}}|\right)|x_{1b}^{\text{freq}}| + \left(1-\frac{1}{2}\lambda_b|x_{2b}^{\text{freq}}|\right)|x_{2b}^{\text{freq}}|, \tag{B.42}$$

$$= |x_{1b}^{\text{freq}}| + |x_{2b}^{\text{freq}}| - \frac{1}{2}\lambda_b\left(|x_{1b}^{\text{freq}}|^2 + |x_{2b}^{\text{freq}}|^2\right), \tag{B.43}$$

d'où :

$$\frac{1}{2}\lambda_b = \frac{-|x_b^{\text{freq}}| + |x_{1b}^{\text{freq}}| + |x_{2b}^{\text{freq}}|}{|x_{1b}^{\text{freq}}|^2 + |x_{2b}^{\text{freq}}|^2}. \tag{B.44}$$

On remplace dans les formules des transformations pour trouver la solution au problème :

$$\widehat{T}_{1b} = \frac{|x_{2b}^{\text{freq}}|^2 + |x_{1b}^{\text{freq}}|\left(|x_b^{\text{freq}}| - |x_{2b}^{\text{freq}}|\right)}{|x_{1b}^{\text{freq}}|^2 + |x_{2b}^{\text{freq}}|^2}, \tag{B.45}$$

$$\widehat{T}_{2b} = \frac{|x_{1b}^{\text{freq}}|^2 + |x_{2b}^{\text{freq}}|\left(|x_b^{\text{freq}}| - |x_{1b}^{\text{freq}}|\right)}{|x_{1b}^{\text{freq}}|^2 + |x_{2b}^{\text{freq}}|^2}. \tag{B.46}$$

En notation vectorielle le résultat précédent donne :

$$\widehat{\mathbf{T}}_1 = \frac{|\mathbf{x}_2^{\text{freq}}|.^2 + |\mathbf{x}_1^{\text{freq}}| \cdot \left(|\mathbf{x}^{\text{freq}}| - |\mathbf{x}_2^{\text{freq}}|\right)}{|\mathbf{x}_1^{\text{freq}}|.^2 + |\mathbf{x}_2^{\text{freq}}|.^2}, \tag{B.47}$$

$$\widehat{\mathbf{T}}_2 = \frac{|\mathbf{x}_1^{\text{freq}}|.^2 + |\mathbf{x}_2^{\text{freq}}| \cdot \left(|\mathbf{x}^{\text{freq}}| - |\mathbf{x}_1^{\text{freq}}|\right)}{|\mathbf{x}_1^{\text{freq}}|.^2 + |\mathbf{x}_2^{\text{freq}}|.^2}. \tag{B.48}$$

B.5 Séparation : optimisation de la déformation complexe

On rappelle le problème d'optimisation :

$$\min_{\overline{\mathbf{T}}_1, \overline{\mathbf{T}}_2 \in \mathbb{C}^{2B}} \left\|\overline{\mathbf{T}}_1 - \tilde{\mathbf{1}}\right\|^2 + \left\|\overline{\mathbf{T}}_2 - \tilde{\mathbf{1}}\right\|^2 \tag{B.49}$$

$$\text{s.c. } \overline{\mathbf{T}}_1 \cdot \overline{\mathbf{x}}_1^{\text{freq}} + \overline{\mathbf{T}}_2 \cdot \overline{\mathbf{x}}_2^{\text{freq}} = \overline{\mathbf{x}}^{\text{freq}}. \tag{B.50}$$

On réécrit le problème avec les multiplicateurs de Lagrange :

$$\min_{\overline{\mathbf{T}}_1, \overline{\mathbf{T}}_2 \in \mathbb{R}^{2B}} \mathbb{L}(\overline{\mathbf{T}}_1, \overline{\mathbf{T}}_2, \boldsymbol{\lambda}) = \left\|\overline{\mathbf{T}}_1 - \tilde{\mathbf{1}}\right\|^2 + \left\|\overline{\mathbf{T}}_2 - \tilde{\mathbf{1}}\right\|^2 + \sum_{b=1}^{2B} \lambda_b \left(\overline{T}_{1b} \overline{x}_{1b}^{\text{freq}} + \overline{T}_{2b} \overline{x}_{2b}^{\text{freq}} - \tilde{x}_b\right). \tag{B.51}$$

On développe la fonction objectif :

$$\mathbb{L}(\overline{\mathbf{T}}_1, \overline{\mathbf{T}}_2, \boldsymbol{\lambda}) = \sum_{b=1}^{B} \left(\overline{T}_{1b}^2 - 2\overline{T}_{1b} + 1 + \overline{T}_{2b}^2 - 2\overline{T}_{2b} + 1\right) + \sum_{b=B+1}^{2B} \left(\overline{T}_{1b}^2 + \overline{T}_{2b}^2\right) + \sum_{b=1}^{2B} \lambda_b \left(\overline{T}_{1b} \overline{x}_{1b}^{\text{freq}} + \overline{T}_{2b} \overline{x}_{2b}^{\text{freq}} - \overline{x}_b^{\text{freq}}\right). \tag{B.52}$$

On calcule les dérivées partielles :

$$\frac{\partial \mathbb{L}}{\partial \overline{T}_{zb}} = 2\overline{T}_{zb} - 2 + \lambda_b \overline{x}_{zb}^{\text{freq}}, \quad b = 1, ..., B, \tag{B.53}$$

$$\frac{\partial \mathbb{L}}{\partial \overline{T}_{zb}} = 2\overline{T}_{zb} + \lambda_b \overline{x}_{zb}^{\text{freq}}, \quad b = B+1, ..., 2B. \tag{B.54}$$

La résolution donne :

$$\overline{T}_{zb} = 1 - \frac{\lambda_b}{2}\overline{x}_{zb}^{\text{freq}},\ b = 1,...,B, \tag{B.55}$$

$$\overline{T}_{zb} = -\frac{\lambda_b}{2}\overline{x}_{zb}^{\text{freq}},\ b = B+1,...,2B. \tag{B.56}$$

On utilise la contrainte pour trouver λ_b pour $b = 1,...,B$:

$$\overline{x}_b^{\text{freq}} = \overline{T}_{1b}\overline{x}_{1b}^{\text{freq}} + \overline{T}_{2b}\overline{x}_{2b}^{\text{freq}}, \tag{B.57}$$

$$\overline{x}_b^{\text{freq}} = \left(1 - \frac{\lambda_b}{2}\overline{x}_{1b}^{\text{freq}}\right)\overline{x}_{1b}^{\text{freq}} + \left(1 - \frac{\lambda_b}{2}\overline{x}_{2b}^{\text{freq}}\right)\overline{x}_{2b}^{\text{freq}}, \tag{B.58}$$

$$\overline{x}_b^{\text{freq}} = \overline{x}_{1b}^{\text{freq}} - \frac{\lambda_b}{2}\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \overline{x}_{2b}^{\text{freq}} - \frac{\lambda_b}{2}\left(\overline{x}_{2b}^{\text{freq}}\right)^2, \tag{B.59}$$

$$\Rightarrow \frac{\lambda_b}{2} = \frac{\overline{x}_{1b}^{\text{freq}} + \overline{x}_{2b}^{\text{freq}} - \overline{x}_b^{\text{freq}}}{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \left(\overline{x}_{2b}^{\text{freq}}\right)^2}. \tag{B.60}$$

On utilise la contrainte pour trouver λ_b pour $b = B+1,...,2B$:

$$\overline{x}_b^{\text{freq}} = \overline{T}_{1b}\overline{x}_{1b}^{\text{freq}} + \overline{T}_{2b}\overline{x}_{2b}^{\text{freq}}, \tag{B.61}$$

$$\overline{x}_b^{\text{freq}} = -\frac{\lambda_b}{2}\left(\overline{x}_{1b}^{\text{freq}}\right)^2 - \frac{\lambda_b}{2}\left(\overline{x}_{2b}^{\text{freq}}\right)^2, \tag{B.62}$$

$$\Rightarrow \frac{\lambda_b}{2} = -\frac{\overline{x}_b^{\text{freq}}}{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \left(\overline{x}_{2b}^{\text{freq}}\right)^2}. \tag{B.63}$$

On obtient ainsi l'estimation des déformations :

$$\widehat{\overline{T}}_{1b} = \frac{\left(\overline{x}_{2b}^{\text{freq}}\right)^2 - \overline{x}_{1b}^{\text{freq}}\overline{x}_{2b}^{\text{freq}} + \overline{x}_b^{\text{freq}}\overline{x}_{1b}^{\text{freq}}}{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \left(\overline{x}_{2b}^{\text{freq}}\right)^2},\quad b = 1,...,B, \tag{B.64}$$

$$\widehat{\overline{T}}_{2b} = \frac{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 - \overline{x}_{1b}^{\text{freq}}\overline{x}_{2b}^{\text{freq}} + \overline{x}_b^{\text{freq}}\overline{x}_{2b}^{\text{freq}}}{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \left(\overline{x}_{2b}^{\text{freq}}\right)^2},\quad b = 1,...,B, \tag{B.65}$$

$$\widehat{\overline{T}}_{1b} = \frac{\overline{x}_b^{\text{freq}}\overline{x}_{1b}^{\text{freq}}}{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \left(\overline{x}_{2b}^{\text{freq}}\right)^2},\quad b = B+1,...,2B, \tag{B.66}$$

$$\widehat{\overline{T}}_{2b} = \frac{\overline{x}_b^{\text{freq}}\overline{x}_{2b}^{\text{freq}}}{\left(\overline{x}_{1b}^{\text{freq}}\right)^2 + \left(\overline{x}_{2b}^{\text{freq}}\right)^2},\quad b = B+1,...,2B. \tag{B.67}$$

En notation vectorisée cela donne :

$$\widehat{\mathrm{Re}(\mathbf{T}_1)} = \frac{\mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right) \cdot \mathrm{Re}\left(\mathbf{x}^{\mathrm{freq}} - \mathbf{x}_2^{\mathrm{freq}}\right)}{\mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right)^{\cdot 2}}, \tag{B.68}$$

$$\widehat{\mathrm{Re}(\mathbf{T}_2)} = \frac{\mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right) \cdot \mathrm{Re}\left(\mathbf{x}^{\mathrm{freq}} - \mathbf{x}_1^{\mathrm{freq}}\right)}{\mathrm{Re}\left(\mathbf{x}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Re}\left(\mathbf{x}_2^{\mathrm{freq}}\right)^{\cdot 2}}, \tag{B.69}$$

$$\widehat{\mathrm{Im}(\mathbf{T}_1)} = \frac{\mathrm{Im}\left(\mathbf{x}^{\mathrm{freq}}\right) \cdot \mathrm{Im}\left(\mathbf{x}_1^{\mathrm{freq}}\right)}{\mathrm{Im}\left(\overline{\mathbf{x}}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Im}\left(\overline{\mathbf{x}}_2^{\mathrm{freq}}\right)^{\cdot 2}}, \tag{B.70}$$

$$\widehat{\mathrm{Im}(\mathbf{T}_2)} = \frac{\mathrm{Im}\left(\mathbf{x}^{\mathrm{freq}}\right) \cdot \mathrm{Im}\left(\mathbf{x}_2^{\mathrm{freq}}\right)}{\mathrm{Im}\left(\overline{\mathbf{x}}_1^{\mathrm{freq}}\right)^{\cdot 2} + \mathrm{Im}\left(\overline{\mathbf{x}}_2^{\mathrm{freq}}\right)^{\cdot 2}}. \tag{B.71}$$

Annexe C

Autres résultats pour la séparation

Cette annexe regroupe les résultats de séparation pour les métriques SIR et SAR, à la fois sur l'ensemble d'entrainement et l'ensemble de test.

C.1 Résultats sur l'ensemble d'apprentissage

Les Tableaux C.1 et C.3 comportent respectivement les scores SIR et SAR sur l'ensemble d'apprentissage pour les méthodes oracles. Les Tableaux C.2 et C.4 comportent respectivement les scores SIR et SAR sur l'ensemble d'apprentissage pour les différentes méthodes considérées.

C.2 Résultats sur l'ensemble de test

Les Tableaux C.5 et C.7 comportent respectivement les scores SIR et SAR sur l'ensemble de test pour les méthodes oracles. Les Tableaux C.6 et C.8 comportent respectivement les scores SIR et SAR sur l'ensemble de test pour les différentes méthodes considérées.

Tableau C.1 – Tâche de séparation sur l'ensemble d'apprentissage. Score SIR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les oracles. Plus le score SIR est élevé meilleur est le résultat.

Méthode	Param.	SIR	
		Voix	Détonation
MIX	-	4,6 (5,5)	-4,2 (5,3)
IBM	1	23,8 (4,5)	20,7 (3,9)
	2	27,1 (4,3)	23,7 (3,7)
IRM	1	22,9 (4,9)	16,9 (3,6)
	2	25,3 (4,4)	22,0 (3,7)

Tableau C.2 – Tâche de séparation sur l'ensemble d'apprentissage. Score SIR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les différentes méthodes : la méthode RASE (déclinée suivant les deux propositions DM-GMM et Def-MAP respectivement), la méthode RARE, ainsi que les méthodes concurrentes NMF et PLCA (non supervisée et supervisée respectivement) et le MSonyNet (réseau de neurones). Plus le score SIR est élevé meilleur est le résultat.

Méthode		Param.	SIR	
			Voix	Détonation
			Min. 4,6 Max. 27,1	Min. -4,2 Max. 23,7
RASE	DM-GMM	10 comp.	12,2 (6,0)	9,6 (6,2)
		50 comp.	14,0 (6,0)	7,9 (5,8)
		100 comp.	14,2 (5,6)	8,4 (6,3)
RASE	Def-MAP	$\alpha = 10^{-4}$	15,6 (5,8)	0,1 (5,3)
		$\alpha = 10^{-3}$	18,2 (5,8)	2,9 (5,4)
		$\alpha = 10^{-2}$	20,1 (5,1)	14,8 (6,0)
		$\alpha = 10^{-1}$	20,4 (4,6)	19,3 (4,2)
		$\alpha = 1$	20,4 (4,6)	19,1 (4,2)
		$\alpha = 10$	20,4 (4,6)	19,1 (4,1)
RARE	-	-	15,2 (5,9)	3,7 (5,8)
NMF	Non sup	2 comp.	6,9 (5,8)	-3,9 (5,9)
	Sup	100 comp.	9,9 (5,4)	14,4 (6,2)
PLCA	Non sup	2 comp.	6,1 (5,9)	-4,3 (5,6)
	Sup	100 comp.	11,2 (5,7)	7,4 (5,6)
MSonyNet	-	-	16,9 (4,8)	19,3 (5,3)

Tableau C.3 – Tâche de séparation sur l'ensemble d'apprentissage. Score SAR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les oracles. Plus le score SAR est élevé meilleur est le résultat.

Méthode	Param.	SAR	
		Voix	Détonation
IBM	1	14,8 (3,7)	9,8 (3,8)
	2	14,8 (3,7)	9,7 (3,7)
IRM	1	15,8 (3,6)	11,1 (4,0)
	2	15,9 (3,7)	11,1 (4,0)

Tableau C.4 – Tâche de séparation sur l'ensemble d'apprentissage. Score SAR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les différentes méthodes : la méthode RASE (déclinée suivant les deux propositions DM-GMM et Def-MAP respectivement), la méthode RARE, ainsi que les méthodes concurrentes NMF et PLCA (non supervisée et supervisée respectivement) et le MSonyNet (réseau de neurones). Plus le score SDR est élevé meilleur est le résultat.

Méthode		Param.	SAR	
			Voix	Détonation
			Min. 0 Max. 15,9	Min. 0 Max. 11,1
RASE	DM-GMM	10 comp.	11,5 (3,6)	4,7 (4,0)
		50 comp.	10,7 (3,9)	4,9 (4,1)
		100 comp.	10,9 (4,3)	5,4 (3,9)
RASE	Def-MAP	$\alpha = 10^{-4}$	7,0 (2,8)	8,2 (3,6)
		$\alpha = 10^{-3}$	8,0 (3,3)	6,8 (3,9)
		$\alpha = 10^{-2}$	13,6 (4,2)	8,4 (4,2)
		$\alpha = 10^{-1}$	14,7 (3,8)	9,1 (4,2)
		$\alpha = 1$	14,7 (3,6)	9,0 (4,2)
		$\alpha = 10$	14,7 (3,6)	9,0 (4,2)
RARE	-	-	8,5 (2,3)	5,5 (3,1)
NMF	Non sup	2 comp.	8,6 (7,1)	7,1 (2,6)
	Sup	100 comp.	13,6 (3,6)	4,8 (3,5)
PLCA	Non sup	2 comp.	8,4 (2,2)	7,5 (2,6)
	Sup	100 comp.	12,0 (3,2)	5,4 (3,3)
MSonyNet	-	-	10,8 (5,0)	5,2 (5,5)

Tableau C.5 – Tâche de séparation sur l'ensemble de test. Score SIR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les oracles. Plus le score SIR est élevé meilleur est le résultat.

Méthode	Param.	SIR	
		Voix	Détonation
MIX	-	4,9 (4,2)	-4,6 (4,2)
IBM	1	24,2 (4,0)	21,2 (3,8)
	2	27,2 (4,4)	24,4 (3,6)
IRM	1	23,2 (4,4)	17,3 (3,3)
	2	25,6 (4,3)	22,5 (3,6)

Tableau C.6 – Tâche de séparation sur l'ensemble de test. Score SIR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les différentes méthodes : la méthode RASE (déclinée suivant les deux propositions DM-GMM et Def-MAP respectivement), la méthode RARE, ainsi que les méthodes concurrentes NMF et PLCA (non supervisée et supervisée respectivement) et le MSonyNet (réseau de neurones). Plus le score SIR est élevé meilleur est le résultat.

Méthode		Param.	SIR	
			Voix	Détonation
			Min. 4,9 Max. 27,2	Min. -4,6 Max. 24,4
RASE	DM-GMM	10 comp.	13,0 (4,6)	9,4 (6,4)
		50 comp.	14,9 (4,9)	8,3 (6,3)
		100 comp.	14,9 (4,1)	9,1 (6,2)
RASE	Def-MAP	$\alpha = 10^{-4}$	11,7 (4,7)	-3,7 (4,7)
		$\alpha = 10^{-3}$	12,6 (4,7)	-3,4 (4,6)
		$\alpha = 10^{-2}$	13,2 (4,7)	-2,4 (5,1)
		$\alpha = 10^{-1}$	13,1 (4,4)	-2,0 (5,1)
		$\alpha = 1$	13,0 (4,4)	-2,0 (5,1)
		$\alpha = 10$	13,0 (4,3)	-2,0 (5,1)
RARE	-	-	13,0 (4,2)	-0,9 (5,6)
NMF	Non sup	2 comp.	7,3 (4,6)	-4,3 (4,8)
	Sup	100 comp.	10,1 (4,1)	11,9 (6,6)
PLCA	Non sup	2 comp.	6,6 (4,7)	-4,7 (4,6)
	Sup	100 comp.	11,5 (4,1)	6,6 (5,4)
MSonyNet	-	-	17,5 (4,4)	20,2 (4,6)

Tableau C.7 – Tâche de séparation sur l'ensemble de test. Score SAR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les oracles. Plus le score SAR est élevé meilleur est le résultat.

Méthode	Param.	SAR	
		Voix	Détonation
IBM	1	16,0 (3,4)	9,5 (3,2)
	2	15,8 (3,4)	9,5 (3,1)
IRM	1	17,1 (3,5)	10,8 (3,4)
	2	17,3 (3,6)	10,8 (3,4)

Tableau C.8 – Tâche de séparation sur l'ensemble de test. Score SAR moyen (et écart-type) en dB pour la séparation de Voix et de Détonation pour les différentes méthodes : la méthode RASE (déclinée suivant les deux propositions DM-GMM et Def-MAP respectivement), la méthode RARE, ainsi que les méthodes concurrentes NMF et PLCA (non supervisée et supervisée respectivement) et le MSonyNet (réseau de neurones). Plus le score SAR est élevé meilleur est le résultat.

Méthode		Param.	SAR	
			Voix	Détonation
			Min. 0 Max. 17,3	Min. 0 Max. 10,8
RASE	DM-GMM	10 comp.	11,9 (2,7)	4,4 (3,4)
		50 comp.	11,1 (3,1)	4,8 (3,7)
		100 comp.	11,6 (3,5)	5,3 (3,6)
RASE	Def-MAP	$\alpha = 10^{-4}$	3,1 (3,5)	9,3 (4,2)
		$\alpha = 10^{-3}$	3,7 (2,6)	9,7 (3,3)
		$\alpha = 10^{-2}$	4,2 (2,1)	7,5 (2,9)
		$\alpha = 10^{-1}$	4,4 (2,0)	7,1 (2,7)
		$\alpha = 1$	4,4 (2,0)	7,1 (2,8)
		$\alpha = 10$	4,4 (2,0)	7,1 (2,7)
RARE	-	-	5,6 (1,8)	6,0 (2,2)
NMF	Non sup	2 comp.	8,3 (2,2)	6,4 (2,0)
	Sup	100 comp.	14,2 (3,0)	4,7 (3,0)
PLCA	Non sup	2 comp.	8,1 (1,6)	7,0 (1,8)
	Sup	100 comp.	12,0 (2,3)	4,9 (2,7)
MSonyNet	-	-	12,3 (4,0)	6,2 (5,3)

Annexe D

Implémentation numérique des méthodes

De part l'objectif de traitement de l'audio en temps-réel, les méthodes développées dans la thèse ont été implémentées et optimisées au niveau algorithmique mais également informatique. Cette annexe présente des éléments de codes pour notre méthode mais également pour certaines méthodes concurrentes. La majeure partie a été réalisée sur MATLAB et quelques parties sur Python.

D.1 Classification en temps-réel

D.1.1 RARE : structure des données

Les données brutes étant des sons, ils sont stockés en mémoire sous la forme de fichier audio .wav ou .mp3. La première étape a consisté à trouver un format de fichiers plus adapté pour les traitements liés à la classification. Pour une base de données particulière, j'ai donc d'abord réalisé un découpage de tous les sons en frames de taille Δt décalé de δt et j'ai stocké ces frames dans une matrice `database` $\in \mathbb{R}^{n\times\Delta t+2}$ dont chaque ligne est formée par la frame, l'indice k de la classe ainsi que l'indice du son.

La séparation entre données d'apprentissage et de test a ensuite été réalisée avec un schéma de validation croisée à 5 folds. Pour les différents folds, les parties apprentissage et test sont stockées dans une structure MATLAB (ou un dictionnaire Python) de la forme `database{fold}.train` et `database{fold}.test`.

Le calcul des spectres de puissances normalisés a également été stocké sous une forme similaire, c'est-à-dire une matrice `feature`$\in \mathbb{R}^{n\times B+2}$ pour tous les spectres de puissances normalisés et `feature{fold}.train` et `feature{fold}.test` pour les différents folds apprentissage et test.

D.1.2 RARE : optimisation des calculs

Une première optimisation consiste à travailler avec les paramètres des noyaux $\mathbf{p}_i^{(q)}$ en log :

```
1 log_p = gpuArray(log(p));
```

Comme explicité dans la section sur la réduction de modèle, la plupart des calculs sont des multiplications de matrices élément par élément :

```
% Quantification du spectre
x = gpuArray(q * x ./ sum(x));

% Evaluation de chaque noyau
rx               = repmat(x, [n_kernels,1]);
evaluated_kernel = sum(rx .* log_p,2);

% Calcul des distributions
for z = 1:K
        evaluated_kernel_z = evaluated_kernel(msize(z)+1:msize(z+1));
        prob_dist(z)       = LSE(gather(evaluated_kernel_z));
end
prob_dist = prob_dist - log(model_size);
prob_dist = LSE_normprob(gather(prob_dist + log_prior));
```

De part la structure des opérations, l'utilisation d'un GPU diminue considérablement les temps de calcul, d'où l'utilisation des `gpuArray`.

Comme les vraisemblances sont très petites et considérées en échelle log, les `logsumexp` sont utilisés :

```
function out = LSE(x)

[~,M]       = size(x);

[x_max,idx] = max(x,[],2);
out         = x_max + log(sum(exp(x - repmat(x_max,[1,M])),2,'omitnan
    '));
```

```
function out = LSE_normprob(x)

[~,M]       = size(x);
norm_factor = LSE(x);
out         = x - repmat(norm_factor,[1,M]);
```

D.1.3 Implémentation des réseaux de neurones

Les différents réseaux de neurones considérés dans cette thèse ont été implémentés en Python avec la librairie Keras. L'avantage principal de cette librairie est sa simplicité d'installation et d'utilisation. En effet, un réseau de neurones est créé en déclarant séquentiellement les différentes couches qui le composent. Dans l'exemple suivant, le réseau DCNN de Piczak [85] est déclaré en Python, en déclarant successivement les différentes couches convolutives `Conv2D` qui prennent en paramètre le nombre de *feature maps* (`filters`) ainsi que leur dimension (`kernel_size`) et la fonction d'activation (`activation`). Les couches de pooling sont déclinées suivant le nombre de dimension dans le réseaux (ici 2D d'où le nom `MaxPooling2D`). Au final des couches denses sont ajoutées (`Dense`) avec la dernière qui est une couche softmax pour réaliser la classification. Le réseau est optimisé avec l'optimisation `Adam` [51], qui est une variante de la SGD avec correction du bias.

```
model = Sequential()

model.add(Conv2D(filters = 80, kernel_size=(57,6), activation='relu',
    input_shape=(2,nBand,nFrames), data_format='channels_first'))
model.add(MaxPooling2D(pool_size=(4,1), strides=(1,1), data_format='
    channels_first'))
model.add(Dropout(0.5))

model.add(Conv2D(filters = 80, kernel_size = (1,3), activation='relu'
    , data_format='channels_first'))
model.add(MaxPooling2D(pool_size=(1,3), strides=(1,3),  data_format='
    channels_first'))

model.add(Reshape(target_shape=(800,)))

model.add(Dense(5000, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(5000, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(nClasses, activation='softmax'))

model.compile(optimizer='Adam', loss='categorical_crossentropy',
    metrics=['accuracy'])
```

Le CRNN [18] est créé en utilisant la même librairie en utilisant le code ci-dessous. La différence avec le DCNN est la présence des couches `BatchNormalization` entre les convolutions et les poolings. De plus, la fonction de perte du CRNN est l'entropie croisée binaire (`binary_crossentropy`) au lieu de l'entropie croisée catégorielle car le réseau est entrainé pour une classification multi-label.

```
model = Sequential()

model.add(Conv2D(256, 5, activation='relu', input_shape=(1,F,nFrames)
    , data_format='channels_first', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(5,1), strides=None, padding='valid'
    , data_format='channels_first'))
model.add(Dropout(0.25))

model.add(Conv2D(256, 5, activation='relu', data_format='
    channels_first', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(4,1), strides=None, padding='valid'
    , data_format='channels_first'))
model.add(Dropout(0.25))

model.add(Conv2D(256, 5, activation='relu', data_format='
    channels_first', padding='same'))
```

```
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,1), strides=None, padding='valid'
    , data_format='channels_first'))
model.add(Dropout(0.25))

model.add(Reshape((256,nFrames)))
model.add(Permute((2,1)))

model.add(GRU(256, return_sequences = True))
model.add(Dropout(0.25))

model.add(Dense(nClasses, activation='sigmoid'))
model.add(BatchNormalization())

model.add(Permute((2,1)))

model.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['
    accuracy'])
```

D.1.4 Mélange de Dirichlet

Ne trouvant pas de librairie standard pour estimer un mélange de Dirichlet, j'ai dû coder grâce à l'article de Ma *et al.* [63] cette estimation. Celle-ci utilise l'algorithme EM ainsi que une optimisation de Newton-Raphson (NR) pour estimer les paramètres des distributions de Dirichlet.

```
function [propEM, alphaNR, L] = fit_mixture_dirichlet(x, param)

% Number of components
M     = param.M;
% Number of samples and dimension
[N,D] = size(x);

% Initiale parameters values
propEM      = ones(M,1) ./ M;
alphaFactor = param.alphaFactor;
alphaNR     = alphaFactor*rand(M,D);

% Number of iterations of the EM algorithm
iterMax   = param.iterMax;
% Log-likelihood
L         = zeros(1,iterMax);
% Number of iterations of the NR algorithm
iterNRMax = param.iterNRMax;
% Main loop
for iter = 1:iterMax
        disp(['iter: ',num2str(iter), ' / ',num2str(iterMax)])
        % E-step
        auxDir          = zeros(N,M); % log dirichlet proba
```

```
        for m = 1:M
                auxDir(:,m) = dirichlet_pdf(x, alphaNR(m,:));
        end
        z       = exp(LSE_normprob( repmat(log(propEM)',[N,1]) +
            auxDir ));
        idx_nan = find(~isnan(z(:,1)));
        % M-step
        propEM = mean(z(idx_nan,:))';
        % NR loop
        for iter_nr = 1:iterNRMax
                for m = 1:M
                        % Jacobian of the objective function
                        Jf = diag(psi(1,alphaNR(m,:))) - psi(1,sum(
                            alphaNR(m,:))) * ones(D,D);
                        % Objective function
                        f  = psi(alphaNR(m,:)) - psi(sum(alphaNR(m,:)
                            )) - sum(repmat(z(idx_nan,m),[1,D]) .* log
                            (x(idx_nan,:)),1) ./ sum(z(idx_nan,m));
                        % Update
                        alphaNR(m,:) = alphaNR(m,:) - (Jf \ f')';
                end
                % In case of negative value, picka random positive
                    value
                alphaNR(alphaNR < 0) = rand(1);
                end
                if M == 1 % For a single Dirichlet distribution
                        L(iter) = dirichlet_likelihood(x, alphaNR);
                else      % For a mixture
                        L(iter) = mixture_dirichlet_likelihood(x,
                            alphaNR, propEM);
                end
                if isnan(L(iter)) % If divergence, stop the algorithm
                        L = L(1:iter);
                        break;
                end
        end
end
```

D.2 Séparation en temps-réel

D.2.1 RASE : structure des données

De manière à calculer efficacement les différentes estimations de séparation (notre méthode et les méthodes concurrentes), j'ai utilisé un paradigme particulier utilisant les structures MATLAB. La variable `inputRASE` contient les différentes variables utilisées pour la séparation à la frame `t` dont le spectre à séparer `spectrum`, la base d'apprentissage des spectres complexes `database`, le nombre de spectres dans la base `nBloc` ainsi que les probabilités a priori sur les sons `probSounds`. Le résultat de la séparation est stocké dans `outputsRase` et contient notamment le spectre estimé

par agrégation des probabilités `estSpect` et par le MAP `estSpectMAP`, ainsi que la fonction de coût `loss` et l'indice des sons sélectionnés par le MAP `idxMAP`.

D.2.2 RASE : optimisation des calculs

De même que pour RARE, la séparation étudiée dans la thèse a été codée et optimisée pour être rapide d'exécution. La fonction suivante est utilisée pour l'estimation d'un spectre de la frame `t`.

```
function outputsRase = separateRASE(inputRASE)

% Get input variables
% Input mixed spectrum
spectrum      = inputRASE.spectrum;
%  Learning set of complex spectrum
database      = inputRASE.database;
% Number of spectra
nBloc         = inputRASE.nBloc;
% Prior probabilities of sounds
probSounds    = inputRASE.probSounds;
% Frame index
t             = inputRASE.t;

% Initialize relevant variables and precomputed spectra
nClasses      = length(database);
F             = size(database{1},1);
r_spectrum    = repmat(spectrum,[1,nBloc(2)]);
r_angle       = repmat(exp(1i * angle(spectrum)),[1,nClasses]);
dimReshape    = [F,1,nClasses];

estV          = zeros([F,nBloc]);
loss          = zeros(nBloc);

% Compute each pair of deformation
for j = 1:nBloc(1)
        % Spectra from class 1
        x_1 = database{1}(:,j,t);
        x_1 = repmat(x_1,[1,nBloc(2)]);
        % Spectra from class 2
        x_2 = database{2}(:,:,t);
        % Transformation
        vol     = zeros(2*F,nBloc(2));
        aux     = zeros(2*F,nBloc(2));

        % Optimize norm(vol - id)
        % Vectorize complex variables
        tilde_x   = [real(r_spectrum);imag(r_spectrum)];
        tilde_x_1 = [real(x_1);imag(x_1)];
        tilde_x_2 = [real(x_2);imag(x_2)];
        sum_x_s   = tilde_x_1.^2 + tilde_x_2.^2 + eps;
```

```
42          % Update transformation
43          aux_vol_1(1:F,:)     = (tilde_x_2(1:F,:).^2 - tilde_x_1(1:F
                ,:) .* tilde_x_2(1:F,:) + tilde_x(1:F,:) .* tilde_x_1(1:F
                ,:)) ./ sum_x_s(1:F,:);
44          aux_vol_1(F+1:2*F,:) = tilde_x(F+1:2*F,:) .* tilde_x_1(F+1:2*
                F,:) ./ sum_x_s(F+1:2*F,:);
45          aux_vol_2(1:F,:)     = (tilde_x_1(1:F,:).^2 - tilde_x_1(1:F
                ,:) .* tilde_x_2(1:F,:) + tilde_x(1:F,:) .* tilde_x_2(1:F
                ,:)) ./ sum_x_s(1:F,:);
46          aux_vol_2(F+1:2*F,:) = tilde_x(F+1:2*F,:) .* tilde_x_2(F+1:2*
                F,:) ./ sum_x_s(F+1:2*F,:);
47          vol(1:F,:)           = aux_vol_1(1:F,:) + 1i * aux_vol_1(F
                +1:2*F,:);
48          vol(F+1:2*F,:)       = aux_vol_2(1:F,:) + 1i * aux_vol_2(F
                +1:2*F,:);
49
50          % Apply transformation
51          aux           = [x_1;x_2] .* vol;
52          % Stock estimated spectra of class 1 and 2
53          estV(:,j,:,1) = aux(1:F,:);
54          estV(:,j,:,2) = aux(F+1:2*F,:);
55          % Compute loss function
56          loss(j,:)     = sum(conj(vol - 1) .* (vol - 1));
57  end
58  % Compute probabilities
59  logProbCond = -loss;
60  probCond    = checkNanProb(logProbCond);
61
62  estSpect    = zeros(F,nClasses);
63  prob        = zeros(nBloc);
64
65  % Aggregate probabilities
66  inputUpdate.probCond = probCond;
67  inputUpdate.probSounds = probSounds;
68  for j = 1:nBloc(1)
69          for k = 1:nBloc(2)
70                  sV                  = squeeze(estV(:,j,k,:));
71
72                  inputUpdate.prob     = prob;
73                  inputUpdate.estSpect = estSpect;
74                  inputUpdate.idx      = [j,k];
75                  inputUpdate.sV       = sV;
76                  [prob,estSpect]      = updateSpectRASE(inputUpdate);
77          end
78  end
79  % Estimated spectra using aggregation of probabilities
80  estSpect    = reshapeSpect(estSpect,r_angle,dimReshape);
81  % Estimated spectra using MAP
82  [row,col]   = findMax(prob,nBloc);
```

```
estSpectMap = squeezeSpect(estV,row,col);
estSpectMap = reshapeSpect(estSpectMap,r_angle,dimReshape);

% Stock output variables
outputsRase.estSpect    = estSpect;
outputsRase.estSpectMap = estSpectMap;
outputsRase.loss        = loss;
outputsRase.idxMAP      = [row,col];
```

D.2.3 PLCA

La PLCA n'étant pas disponible dans des packages standards, j'ai dû la coder à partir de la thèse de Shashanka [99]. Le code a été optimisé aussi bien en mémoire qu'en rapidité d'exécution en utilisant les propriétés de MATLAB (réduire au maximum les boucles, utiliser des multiplications de matrices, *etc.*)

```
function [P_z,P_f,P_t] = plca(V,Z,iterMax,fixed)

% Number of time-frequency bins
[F,T]   = size(V);

% each component can be estimated (for learning) or fixed (for
    testing)
if isempty(fixed{1})
P_z = rand(Z,1);
P_z = P_z ./ sum(P_z);
else
P_z = fixed{1};
end
if isempty(fixed{2})
P_f = rand(Z,F);
P_f = P_f ./ repmat(sum(P_f,2),[1,F]);
else
P_f = fixed{2};
end
if isempty(fixed{3})
P_t = rand(Z,T);
P_t = P_t ./ repmat(sum(P_t,2),[1,T]);
else
P_t = fixed{3};
end
% Preprocess the spectrogram to accelerate the computation
rV = repmat(reshape(V,[1,F,T]),[Z,1,1]);
% Main loop
for iter = 1:iterMax
% E-Step
% Compute the $P(z|f,t)$
aux   = repmat(reshape((P_z * ones(1,T)) .* P_t,[Z,1,T]),[1,F,1]);
P_zft = aux .* repmat(P_f,[1,1,T]);
```

```
P_zft = P_zft ./ repmat(sum(P_zft,1),[Z,1,1]);

% M-step
% Compute the product $V_ft P(z|f,t)$
VP    = rV .* P_zft;
% Compute the different sums :
% $\sum_f V_ft P(z|f,t)$
VP2   = reshape(sum(VP,2),[Z,T]);
% $\sum_t V_ft P(z|f,t)$
VP3   = sum(VP,3);
% $\sum_f \sum_t V_ft P(z|f,t)$
VP32  = sum(VP3,2);
% Update the distributions if they are not fixed
if isempty(fixed{1})
P_z = VP32;
P_z = P_z ./ sum(P_z);
end
if isempty(fixed{2})
P_f = VP3 ./ repmat(VP32,[1,F]);
end
if isempty(fixed{3})
P_t = VP2 ./ repmat(VP32,[1,T]);
end
end
```

Annexe E

Articles de conférences et de journaux

Durant cette thèse, plusieurs articles de conférences et de journaux ont été écrits puis soumis / acceptés. Ces papiers sont regroupés dans cette annexe.

E.1 Articles de conférence.

Le premier papier de conférence [5] a été accepté à la conférence 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) à la Nouvelle-Orléans, Etats-Unis, et est publié dans les proceedings de cette conférence sous la citation :

Maxime Baelde, Christophe Biernacki, Raphaël Greff. A mixture model-based real-time audio sources classification method. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5-9 Mar 2017, pp. 2427-2431, New Orleans, USA.

Le second papier de conférence [6] a été accepté à la conférence 49è Journées de Statistiques à Avignon, France et est publié dans les proceedings de cette conférence sous la citation :

Maxime Baelde, Christophe Biernacki, Raphaël Greff. Classification de signaux audio en temps-réel par un modèle de mélanges d'histogrammes. 49èmes Journées de Statistiques, June 2017, Avignon, France.

E.2 Articles de journaux

Un article de journal a été accepté pour publication dans Pattern Recognition le 21 mars 2019. Il est disponible sous la citation

Maxime Baelde, Christophe Biernacki et Raphaël Greff. "Real-Time monophonic and polyphonic audio classification from power spectra". In : Pattern Recognition 92 (august 2019), p. 82-92.

A MIXTURE MODEL-BASED REAL-TIME AUDIO SOURCES CLASSIFICATION METHOD

Maxime Baelde[1,2], Christophe Biernacki[1], Raphaël Greff[2]

[1]University Lille 1, CNRS, Inria
[2]A-Volute

ABSTRACT

Recent research on machine learning focuses on audio source identification in complex environments. They rely on extracting features from audio signals and use machine learning techniques to model the sound classes. However, such techniques are often not optimized for a real-time implementation and in multi-source conditions. We propose a new real-time audio single-source classification method based on a dictionary of sound models (that can be extended to a multi-source setting). The sound spectrums are modeled with mixture models and form a dictionary. The classification is based on a comparison with all the elements of the dictionary by computing likelihoods and the best match is used as a result. We found that this technique outperforms classic methods within a temporal horizon of 0.5s per decision (achieved 6% of errors on a database composed of 50 classes). Future works will focus on the multi-sources classification and reduce the computational load.

Index Terms— real-time, audio identification, statistical learning, mixture models, sound classification.

1. INTRODUCTION

Audio source classification is a vast and trendy topic in machine learning, and can be divided into three groups. Music Information Retrieval (MIR) aims at recognizing musical instruments [1] or musical genres [2] from musics. Automatic Speech Recognition (ASR) aims at detecting and identifying speakers in audio recordings [3]. Environmental Sound Recognition (ESR) aims at recognizing classes of sounds that are neither music nor speech [4]: for instance, airplanes or gunshots.

Typical audio source classification methods include two stages. The first one consists in extracting features from the signals using audio descriptors [5]. Common features are temporal (energy, zero crossing rate,...), spectral (Mel Frequency Cepstral Coefficient, centroid,...) or harmonic (fundamental frequency, inharmonicity,...). This step extracts relevant information from the signal and can be seen as a dimension reduction. The second one uses machine learning algorithm to model the sound classes based on the previous features. Commonly used algorithms are Gaussian Mixture Models (GMM) [6, 7], Support Vector Machines (SVM) [8, 9], Hidden Markov Models (HMM) [10, 11] or Neural Networks (NN) [12, 13]. More recently, research focuses on neural networks and uses them for features extraction and classification at the same time. These networks often involve convolutional layers and deep architecture (Deep Convolutional Neural Network, DCNN) to model fine details in signals [3, 14].

In this study, we develop an audio single-source classification system based on a dictionary of models in a probabilistic framework, using the mixture model theory [15]. Indeed, the future goal of this research is to identify several sound sources at the same time, which is merely impossible with the current techniques. The use of mixture models allows to deal with mixture of sounds and therefore to identify simultaneous audio sources. Each sound spectrum is modeled by a mixture model and all the models constitute a dictionary. The classification is performed by comparing an unknown signal with the elements of the dictionary by computing likelihoods, aggregating these probabilities and taking the Maximum A Posteriori (MAP).

The rest of the paper is organized as follows. The creation of the dictionary is presented in Section 2. The single-source identification procedure that uses the previous dictionary is detailed in Section 3. The experiments carried out to assess the performance of the method are presented in Section 4. Section 5 presents the results of the experiments. Finally, a brief discussion is presented in Section 6 and concludes the paper.

2. CREATION OF THE DICTIONARY

The classification algorithm deals with multiple classes of sounds, denoted G_i, $i = 1, ..., I$, and a mono-channel stream. For instance, the classes can be $G_1 = \text{airplane}$, $G_2 = \text{gunshot}$. In each group G_i there are multiple sounds, labeled C_{ij}, $j = 1, ..., J_i$. For instance in G_1, $C_{11} = \text{airplane\_1}$, and in G_2, $C_{23} = \text{gunshot\_3}$. As a typical signal processing step for real-time application, the sounds C_{ij} are splitted into buffers, labeled c_{ijk}, $k = 1, ..., K_{ij}$ of size T samples with a time shift of D samples (see **Fig.** 1). In a signal notation, $\mathbf{c}_{ijk} = \mathbf{C}_{ij}[kD : kD + T]$. In real world applications, the signal does not necessarily fit correctly into the buffer. To tackle this problem, Gaussian white noise is added at the beginning and the end of each sound.

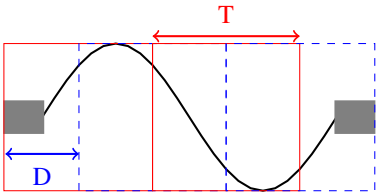


Fig. 1. A signal (black plain line) splitted into buffers of size T samples (red empty square) with a time shift of D samples (blue empty dashed square). Gaussian white noise is added at the beginning and the end (gray full block).

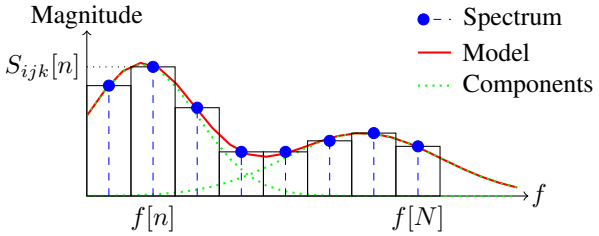


Fig. 2. Modeling of the spectrums as histograms. The spectrum is represented by the blue dots, the mixture model by a red plain line and its components by green dotted lines.

The Fourier spectrum of each buffer is computed, denoted by $\mathbf{s}_{ijk}$. Only the first N bins in the spectrum are kept, because most of the information is included in the low-frequency content of the signal. As our algorithm is designed to classify multiple sounds present at the same time, the energy spectrum is used. Indeed, when two uncorrelated signals are mixed, their energy spectrums are mixed in the same proportion. The spectrums are normalized so that they sum up to N:

$$S_{ijk}[n] = N \frac{\left|s_{ijk}[n]\right|^2}{\sum_{p=1}^{N} \left|s_{ijk}[p]\right|^2}, \tag{2.1}$$

where $|z|$ denotes the modulus of the complex number z. The square values are considered since we want to keep the additivity of the spectrums. These spectrums are considered as histograms and are modeled using a mixture model for binned data [16] (see **Fig. 2**). For the sake of clarity, we will drop the indices (i, j, k) until the end of this section. A mixture model [15] is a mixture of several probability density functions (pdf), the components, which represents the distribution of a random variable. Here the random variable is the frequency f at which the spectrum is computed, and the pdf is parameterized by a set of parameters $\boldsymbol{\theta} = (\pi_m, \mu_m, \sigma_m^2)_{m=1,..,M}$:

$$p\left(f|\boldsymbol{\theta}\right) = \sum_{m=1}^{M} \pi_m \mathcal{N}\left(f\left|\mu_m, \sigma_m^2\right.\right), \tag{2.2}$$

where $p(f|\boldsymbol{\theta})$ means the probability of f parameterized by $\boldsymbol{\theta}$, π_m the mixing coefficients ($\sum_m \pi_m = 1, \pi_m > 0$) and M is the number of components. $\mathcal{N}\left(f\left|\mu_m, \sigma_m^2\right.\right)$ is the density of the univariate normal distribution of mean μ_m and variance σ_m^2, taken at point f:

$$\mathcal{N}\left(f\middle|\mu_m, \sigma_m^2\right) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp\left(-\frac{1}{2}\left(\frac{f-\mu_m}{\sigma_m}\right)^2\right).$$

The frequency f is not known precisely: we know only how many frequencies $S[n]$ fall into the frequency range $\left[f[n], f[n+1]\right[$. This framework is close to the one of McLachlan [16] except that $S[n]$ is not an integer. Thus, given that the fs are independent and identically distributed (i.i.d.), the probability that one sample falls into $\left[f[n], f[n+1]\right[$ is:

$$p\left(S[n]|\boldsymbol{\theta}\right) = \left(\int_{f[n]}^{f[n+1]} p(f|\boldsymbol{\theta}) df\right)^{S[n]}. \tag{2.3}$$

The Expectation-Maximization (EM) algorithm of Dempster [17] is used to estimate the set of parameters $\boldsymbol{\theta}$. This algorithm finds the parameters that maximize the likelihood of the model parameterized by $\boldsymbol{\theta}$. Given that the $S[n]$s are i.i.d., the likelihood is:

$$\mathcal{L}(\boldsymbol{\theta}) = p(\mathbf{S}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p\left(S[n]|\boldsymbol{\theta}\right), \tag{2.4}$$

where $\mathbf{S} = \left[S[1], ..., S[N]\right]$. Starting with an initial parameter estimate $\boldsymbol{\theta}^{(0)}$, the EM algorithm iterates the two following steps until convergence:

(E) Compute: $Q\left(\boldsymbol{\theta}\middle|\boldsymbol{\theta}^{(p)}\right) = \mathbb{E}\left[\log \mathcal{L}(\boldsymbol{\theta})\middle|\mathbf{S}, \boldsymbol{\theta}^{(p)}\right]$,

(M) Maximize: $\boldsymbol{\theta}^{(p+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} Q\left(\boldsymbol{\theta}\middle|\boldsymbol{\theta}^{(p)}\right)$.

This model and the related algorithm require the number of components M in the mixture to be specified. For each buffer, the optimal number of components is chosen by minimizing the Bayesian Information Criterion (BIC) of Schwarz [18]:

$$BIC(M) = -2\log \mathcal{L}\left(\widehat{\boldsymbol{\theta}}\right) + d\log N, \tag{2.5}$$

where $\widehat{\boldsymbol{\theta}}$ denotes the Maximum Likelihood (ML) value of $\boldsymbol{\theta}$ and $d = 3M - 1$ is the dimension of $\boldsymbol{\theta}$, with M varying from 1 to 20 in this case. The estimated parameters are gathered into $\left\{\widehat{\boldsymbol{\theta}}_{ijk}\right\}$, with $k = 1, ..., K_{ij}$, $j = 1, ..., J_i$, $i = 1, ..., I$, which is named the *dictionary*.

3. IDENTIFICATION PROCEDURE

Once the dictionary is computed (this corresponds to the learning step), it can be used to identify unknown sounds. Suppose that at a time r there is a new buffer which is transformed into a normalized energy spectrum $\mathbf{S}^r$. The identification process

consists in finding the correct label G_i^r by aggregating the following probabilities:

1. Compute the likelihoods of all the models of the dictionary $p\left(\mathbf{S}^r \middle| c_{ijk}^r\right) = p\left(\mathbf{S}^r \middle| \widehat{\boldsymbol{\theta}}_{ijk}\right)$, for $k = 1, ..., K_{ij}$, $j = 1, ..., J_i$ and $i = 1, ..., I$ using Eq. (2.4).

2. Compute the likelihood of the sound C_{ij}^r, for $j = 1, ..., J_i$ and $i = 1, ..., I$:

$$p(\mathbf{S}^r|C_{ij}^r) = \sum_{k=1}^{K_{ij}} p(\mathbf{S}^r|c_{ijk}^r)p(c_{ijk}^r|C_{ij}^r). \tag{3.1}$$

3. Compute the likelihood the group G_i^r, for $i = 1, ..., I$:

$$p(\mathbf{S}^r|G_i^r) = \sum_{j=1}^{J_i} p(\mathbf{S}^r|C_{ij}^r)p(C_{ij}^r|G_i^r). \tag{3.2}$$

4. Finally compute the conditional probability of the group G_i^r for $i = 1, ..., I$:

$$p(G_i^r|\mathbf{S}^r) = \frac{p(\mathbf{S}^r|G_i^r)p(G_i^r)}{\sum_h p(\mathbf{S}^r|G_h^r)p(G_h^r)}. \tag{3.3}$$

The decision rule uses R buffers (as in [6]) and is computed as follows. For each buffer r and each class the classifier computes a probability $p(G_i^r|\mathbf{S}^r)$. Then these probabilities are multiplied so as to aggregate the results over a larger temporal horizon:

$$p\left(G_i|\mathbf{S}\right) = \prod_{r=1}^{R} p\left(G_i^r \middle| \mathbf{S}^r\right), \tag{3.4}$$

where $\mathbf{S} = \left[\mathbf{S}^1, ..., \mathbf{S}^R\right]$. The MAP estimate is finally taken on these probabilities: $\widehat{G}_i = \mathrm{argmax}_{G_i}\, p\left(G_i|\mathbf{S}\right)$.

4. EXPERIMENTS

4.1. Setup of the experiments

Several experiments were carried out to assess the quality of our sound sources classification algorithm. Three databases were considered for these experiments. One database (from A-Volute) was composed of 704 video games sounds divided into 9 classes. The ESC databases [19] (50 and 10) were also considered, composed of 2000 environmental sounds from 50 classes for the former and 400 sounds from 10 classes for the latter. Each sound was resampled to 44.1kHz and the mean was subtracted to the signal because it caused instability in the fitting process. Several values were considered for the window size T and the time shift D, mainly $T = [512, 1024, 2048]$ samples and $D = 512$ samples. The number of kept bins N was set to $T/5$ which corresponded to approximately 8kHz in our setting. R was set to 10, that corresponds approximately to 0.5s.

Each dataset were splitted into a training and a test set for cross-validation procedure. We used 80% of the set for training and the remainder for testing. The division was done in a v-fold manner, with $v = 5$. The recognition rate, defined by the number of buffers correctly labeled over the total number of buffers, was used to assess the performance of the system. In a cross-validation setting, a recognition rate for each fold was computed and the cross-validation recognition rate was simply the mean of the v folds recognition rates.

Three probabilities were introduced in Section 3 and have to be specified:

- The probability of a buffer c_{ijk} given a sound C_{ij} was set to 1: $p(c_{ijk}|C_{ij}) = 1$ for $k = 1, ..., K_{ij}$, $j = 1, ..., J_i$ and $i = 1, ..., I$.
- The probability of a sound C_{ij} given a class G_i was uniformly distributed over the class G_i: $p(C_{ij}|G_i) = 1/\#\{G_i\}$ for $j = 1, ..., J_i$ and $i = 1, ..., I$, where $\#\{A\}$ means the number of elements in A.
- The probability of a class G_i was set to the ratio between the number of sounds in G_i and the total number of sounds: $p(G_i) = \#\{G_i\}/\sum_h \#\{G_h\}$ for $i = 1, ..., I$.

4.2. Comparison with other techniques

We compared our algorithm with other state-of-the-art techniques. The first one was a parametric method inspired from Clavel [6]. Acoustic features were extracted from the signals: energy, 8 MFCC, spectral centroid, spectral spread, plus the first and second derivatives. A Principal Component Analysis step was applied on these descriptors and only the first 13 principal components were kept. The classifier was a standard GMM.

The second one was a non-parametric method: a DCNN. A neural network is a set of neurons (computation units) stacked together in multiple layers that can performs classification or regression. The input of a layer l is denoted by $\mathbf{h}_{l-1}$, and the convolutional kernel $\mathbf{W}_l$. The network computes an output $\mathbf{h}_l$ by taking an activation function g: $\mathbf{h}_l = g(\mathbf{W}_l \star \mathbf{h}_{l-1})$, where $\star$ is the convolution operation. Recently, the most popular activation function is the ReLU (Rectified Linear Unit), which is simply $g(\mathbf{X}) = \max(\mathbf{X}, 0)$ element-wise. After a convolutional layer, a max-pooling operation consisting in merging adjacent cells by taking the maximum value is applied. We used the network developed by Piczak [14]. The input was a log-mel spectrogram with its delta, considered as a 2-channel images of size 60×41. The first convolutional layer consisted in 80 filters of size 57×6, with a max-pooling of size 4×3. The second convolutional layer consisted in 80 filters of size 1×3, with a max-pooling of size 1×3. Finally, two fully connected layers composed of 5000 units each and a softmax layer ended the network.

Table 1. The recognition rates in % of the considered methods on the different datasets (A-Volute, ESC-50 and ESC-10).

	A-Volute	ESC-50	ESC-10
Parametric method	73.6	45.5	73.5
Non-parametric method	46.6	53.2	76.0
Our algorithm	96.5	94.0	96.0
Human	91.8	81.3	95.7

For each previous technique, the hyperparameters were adapted so as to have comparable results.

4.3. Human listening tests

Because a human score was available for the ESC dataset [19], we carried out a listening test on the A-Volute dataset. 21 participants were selected and had to classify 10 sounds from the 9 classes of the dataset. It gave a rough estimate of the human good classification rate on this dataset.

5. RESULTS

5.1. Recognition rate

The experiments were carried out for all the values of T. However, only the results for $T = 2048$ are shown because they are the best, and are presented in **Table 1**. The results for the A-Volute's database are the following. The parametric method achieves 73.6% and the non-parametric method 46.6%. Our algorithm outperforms the current methods by achieving a recognition rate of 96.5%. Concerning the ESC-50 database, the parametric method achieves 44%, the non-parametric method 53.2% and our algorithm 94.0%. Finally, on the ESC-10 dataset, the parametric method achieves 73.5%, the non-parametric method 76.0% and our algorithm 96.0%. It is worth noticing that a human can achieve 91.8% on the A-Voute dataset, 94.0% on the ESC-50 database and 95.7% on the ESC-10 database [19]. Our algorithm outperforms state-of-the-art methods and humans on these databases.

5.2. Complexity and execution time

We evaluate the complexity of the previous algorithms at the identification step. Our algorithm is mainly concerned with multiplication. Indeed, by computing the log-probability, computations are just additions and multiplications. All the operations needed to compute the decision for one buffer can be resumed as a $O(\#\{\mathcal{D}\}T/5)$. The parametric method relies mainly on computing exponentials and matrix operations (inversion and determinant), so the complexity is roughly $O(d^3)$ where d is the dimension of the feature vector. The neural network uses convolutions which are the more expensive operations when evaluating the decision. The complexity is thus $O\left(\sum_{l=1}^{L} n_{l-1}(s_l^1 s_l^2)n_l(m_l^1 m_l^2)\right)$, with L the number of layers, n_{l-1} the number of input channels for layer l, s_l^i the dimensions of the input, n_l the number of filters of the layer l and m_l^i the size of the output.

Considering the settings of the experiments, the numbers of operations needed for one buffer for our algorithm are $O(28 \cdot 10^6)$ (A-Volute database), $O(63 \cdot 10^6)$ (ESC-10) and $O(120 \cdot 10^6)$ (ESC-50), for the parametric method $O(2 \cdot 10^3)$ and for the neural network $O(14 \cdot 10^6)$.

We also report the time needed to compute the decision. The machine used a Intel®Core™i7-5820K CPU @3.30GHz. Using a dictionary composed of approximately 70,000 models, it takes 185ms to the system to infer the decision for one buffer. By using an NVidia GeForce®GTX 960 to compute the multiplications, this computational time can be reduced to 35ms.

6. DISCUSSION AND CONCLUSION

The aim of this algorithm is to perform real-time audio multi-source identification. Up to now, this study examined only the single-source case. As we can see in the results, our technique outperforms standard methods (GMM), more recent algorithms (DCNN) and even humans on both benchmark [19] and industrial databases. Despite these good recognition rates, the main advantage of our algorithm is that it can theoretically handle multi-source conditions. The learning algorithm of classic techniques would have to learn every combination of sound classes, which is practically impossible because of the combinatorial and the amount of data required for training. Some research on neural networks begins to study polyphonic identification, as in [20].

The energy spectrum is chosen so as to extend this technique to the multi-source setting. Indeed, we make the assumption that the additivity of two energy spectrums is preserved (uncorrelated sources). Mixture models were employed because they allow a flexible modeling of any signal. Moreover, in a multi-source setting, the signal model is a mixture of mixture, which is well defined in the mixture model framework.

The real-time constraint requires that the signals are not known by advances. Besides, we want the system to have a low latency. This is why such values of T and D were chosen. However, it is not reach yet. Indeed, the larger the dictionary the longer it takes to compute the decision. This is why future work will try to reduce the computation time, for instance by organizing the dictionary in a binary tree.

7. REFERENCES

[1] Cyril Joder, Slim Essid, and Gal Richard, "Temporal Integration for Audio Classification With Application to Musical Instrument Classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 174–186, Jan. 2009.

[2] Gursimran Kour and Neha Mehan, "Music Genre Classification using MFCC, SVM and BPNN," *International Journal of Computer Applications*, vol. 112, no. 6, pp. 12–14, Feb. 2015.

[3] Dimitri Palaz, Mathew Magimai Doss, and Ronan Collobert, "Convolutional Neural Networks-based continuous speech recognition using raw speech signal," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 4295–4299.

[4] Sachin Chachada and C. C. Jay Kuo, "Environmental sound recognition: A survey," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, Oct. 2013, pp. 1–9.

[5] Geoffroy Peeters, Bruno L. Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams, "The Timbre Toolbox: extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, Nov. 2011.

[6] Chlo Clavel, T. Ehrette, and G. Richard, "Events Detection for an Audio-Based Surveillance System," in *2005 IEEE International Conference on Multimedia and Expo*, July 2005, pp. 1306–1309.

[7] Jae-Hun Choi and Joon-Hyuk Chang, "On using acoustic environment classification for statistical model-based speech enhancement," *Speech Communication*, vol. 54, no. 3, pp. 477–490, Mar. 2012.

[8] Sbastien Lecomte, Rgis Lengell, Cdric Richard, Francois Capman, and Bertrand Ravera, "Abnormal events detection using unsupervised One-Class SVM - Application to audio surveillance and evaluation -," in *2011 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, Aug. 2011, pp. 124–129.

[9] Souli Sameh and Lachiri Zied, "On the Use of Time-Frequency Reassignment and SVM-Based Classifier for Audio Surveillance Applications," *International Journal of Image, Graphics and Signal Processing*, vol. 6, no. 12, pp. 17–25, Nov. 2014.

[10] Toni Heittola, Annamaria Mesaros, Antti Eronen, and Tuomas Virtanen, "Audio context recognition using audio event histograms," in *18th European Signal Processing Conference*, Aug. 2010, pp. 1272–1276.

[11] Alberto Bietti, Francis Bach, and Arshia Cont, "An online em algorithm in hidden (semi-)Markov models for audio segmentation and clustering," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 1881–1885.

[12] Robin Biondi, Gareth Dys, Gilles Ferone, Thibault Renard, and Morgan Zysman, "Low Cost Real Time Robust Identification of Impulsive Signals," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 8, no. 9, pp. 1653–1656, 2014.

[13] Cristina P. Dadula and Elmer P. Dadios, "Neural network classification for detecting abnormal events in a public transport vehicle," in *2015 International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, Dec. 2015, pp. 1–6.

[14] Karol J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sept. 2015, pp. 1–6.

[15] Geoffrey McLachlan and David Peel, *Finite Mixture Models*, Wiley, 2000.

[16] G. J. McLachlan and P. N. Jones, "Fitting Mixture Models to Grouped and Truncated Data via the EM Algorithm," *Biometrics*, vol. 44, no. 2, pp. 571–578, 1988.

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[18] Gideon Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, Mar. 1978.

[19] Karol J. Piczak, "ESC: Dataset for Environmental Sound Classification," 2015, pp. 1015–1018, ACM Press.

[20] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen, "Recurrent Neural Networks for Polyphonic Sound Event Detection in Real Life Recordings," *arXiv:1604.00861 [cs]*, Apr. 2016, arXiv: 1604.00861.

Classification de signaux audio en temps-réel par un modèle de mélanges d'histogrammes

Maxime Baelde [1] & Christophe Biernacki [2] & Raphäel Greff [3]

[1,3] *A-Volute,* [1,2] *Université de Lille, CNRS, Inria*
[1] *maxime.baelde@a-volute.com*
[2] *christophe.biernacki@math.univ-lille1.fr*
[3] *raphael.greff@a-volute.com*

Résumé. La reconnaissance sonore consiste à attribuer un label à un signal audio inconnu. Celle-ci repose généralement sur des descripteurs audio ainsi que des modèles d'apprentissage statistique. Néanmoins les modèles actuels peinent à bien classer les sons dans un contexte temps-réel où ces derniers sont hétérogènes. Ce papier propose une nouvelle méthode basée sur un modèle de mélanges d'histogrammes représentant les spectres audio. La reconnaissance consiste à calculer la probabilité de chaque groupe puis à les agréger temporellement. Une étape de réduction du précédent modèle permet par ailleurs de passer au temps-réel. Cette méthode surpasse les algorithmes actuels, et peut atteindre 96,7% de bonne classification sur une base de 50 classes de sons en utilisant 0,5s de données audio.

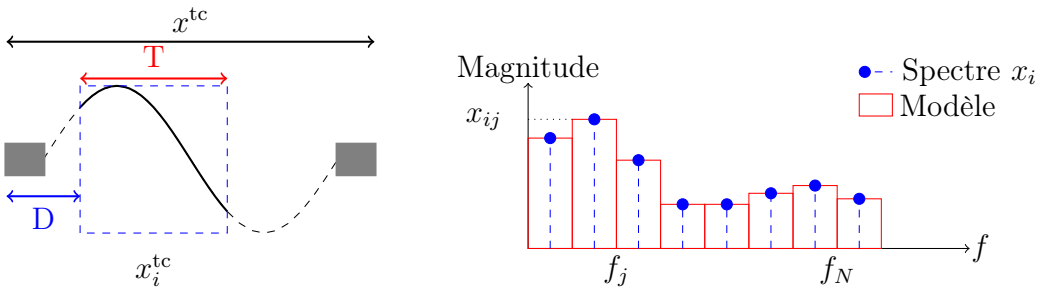
Mots-clés. temps-réel, classification, audio, modèle de mélanges, machine learning.

Abstract. Audio recognition consists in giving a label to an unknown audio signal. It relies on audio descriptors and machine learning algorithms. However, in a real-time context with heterogeneous sounds, the current models lack of performance to classify sounds. This article presents a novel method based on a model of histogram mixture representing audio spectra. The recognition consists in computing the probability of each group and aggregate them temporally. A reduction step of the models allows also to perform this algorithm in real-time. This method outperforms current state-of-the-art algorithms, and achieves an accuracy of 96,7% on a database of 50 classes, using only 0.5s of audio data.

Keywords. real-time, classification, audio, mixture models, machine learning.

1 Introduction

La classification supervisée consiste à attribuer un label à une observation à partir d'un modèle statistique. Appliquée aux technologies de l'audio, elle consiste notamment en la classification de genres musicaux, la reconnaissance vocale ou encore la classification de sons environnementaux



(a) Découpage de x^{tc} en sons partiels x_i^{tc} de taille T avec un décalage D.

(b) Représentation d'un spectre d'énergie x_i par un histogramme.

FIGURE 1 – Modélisation des spectres audio.

Les modèles statistiques de signaux audio reposent en général sur des descripteurs audio (Peeters (2011)) comme l'énergie du signal ou les MFCC (Mel-Frequency Cepstral Coefficients). Les modèles couramment utilisés dans la classification audio sont les mélanges de gaussiennes, les machines à vecteur de support, les modèles de Markov cachés ou les réseaux de neurones convolutifs profonds.

Nous avons décidé de suivre une toute nouvelle approche basée sur des modèles de mélanges (McLachlan (2000)) d'histogramme représentant les spectres audio. C'est une extension du papier de Baelde (2017). L'intérêt de ces modèles est de pouvoir considérer des mélanges de sons qui sont alors modélisés comme des mélanges de mélanges. Une étape de réduction du dictionnaire permet de ne garder que les modèles essentiels et de pouvoir utiliser la technique en temps réel. La classification consiste alors à calculer les probabilités de chaque classe puis à les agréger temporellement. Le contexte dans lequel s'inscrit cet article est le suivant. On dispose de sons répartis en plusieurs classes. L'aspect temps-réel implique que le son n'est jamais connu en entier. C'est pourquoi les sons de chaque classe sont tronqués et décalés de manière à former des sons partiels. De plus, on ajoute du bruit blanc au début et à la fin de chaque son car celui-ci ne commence pas nécessaire au début d'un flux audio, ni ne se termine forcément à la fin.

Nous commençons par présenter dans la Partie 2 la création du modèle, le principe de la classification avec ce dernier ainsi qu'une méthode pour réduire le temps de calcul. La Partie 3 illustre les expériences menées et les résultats. Enfin une discussion ainsi qu'une conclusion sont proposées en Partie 4.

2 Modélisation de spectres sonores par histogrammes

2.1 Ensemble d'apprentissage

L'ensemble d'apprentissage est construit de la manière suivante. On dispose de plusieurs groupes de sons, numérotés $k = 1, ..., g$ (par exemple, le groupe 1 est celui des avions). Dans chaque groupe k, les sons sont numérotés $l = 1, ..., n_k$. Pour le son $x_l^{\text{tc}}(t) \in [-1, 1]$ $(t \in \mathbb{R})$ numéro l continu du domaine temporel, on fabrique les n_{kl} décalages de D unités temporelles et troncatures de taille T unités temporelles, noté $x_{li}^{\text{tc}}(t) = x_l^{\text{tc}}(t \in [iD, iD + T])$ (FIGURE 1(a)). Une unité temporelle est définie comme le pas d'échantillonnage temporel d'un signal sonore. Au total, on obtient $n = \sum_{k=1}^{g} \sum_{l=1}^{n_k} n_{kl}$ sons partiels qui constituent l'ensemble d'apprentissage $(x_i^{\text{tc}}, z_i)_{i=1}^{n}$. Les labels z_i associés à chaque x_i^{tc} indiquent l'appartenance du son partiel au groupe z_i (par exemple $z_i = 1$ indique que x_i^{tc} est un son partiel d'avion). Cependant, en pratique on n'a pas accès aux sons continus x_i^{tc}, qu'il faut discrétiser en $x_i^{\text{td}} \in [-1, 1]^T$ tel que $x_{ij}^{\text{td}} = x_i^{\text{tc}}((j-1)/f_e)$ avec f_e la fréquence d'échantillonnage. De plus, on se place dans le domaine fréquentiel en considérant la transformation en spectre d'énergie normalisée x_i tel que :

$$x_{ij} = \frac{\left| \left(\text{TFD}_T\left(x_i^{\text{td}}\right)\right)_l \right|^2}{\sum_{k=1}^{N} \left| \left(\text{TFD}_T\left(x_i^{\text{td}}\right)\right)_k \right|^2}, \ j = 1, ..., N, \tag{2.1}$$

avec $\text{TFD}_T : \mathbb{R}^T \to \mathbb{C}^T$ l'opérateur de la Transformée de Fourier Discrète (TFD) d'un signal discret sur T points. Ce spectre d'énergie x_i est représenté par un histogramme normalisé (FIGURE 1(b)). On choisit de ne garder que $N < T$ composantes fréquentielles dans le spectre d'énergie car les hautes fréquences ne contiennent pas d'informations pertinentes dans notre cas. De plus, on considère les spectres d'énergies car ils conservent la propriété d'additivité lorsque deux sons décorrélés se mélangent. On se place dans le domaine spectral car le domaine temporel contient trop d'information spécifique à un son et ne permet pas d'extraire de caractéristiques générales communes à un groupe de sons. Les données considérées pour le modèle sont donc les couples $(x_i, z_i)_{i=1}^{n}$.

2.2 Règle de classement

Un nouveau son est traité suivant la même procédure que précédemment pour obtenir l'ensemble de test $(x_j, z_j)_{j=n+1}^{m}$. Dans le cadre de la classification supervisée, on cherche maintenant une règle de classement $R : \mathcal{X} \to \{1, ..., g\}$ qui à un son associe son label : $z = R(x_{n+1}, ..., x_m)$, avec $\mathcal{X} = \underbrace{[0, 1]^N \times ... \times [0, 1]^N}_{m-n}$. On va estimer une règle $\hat{R}$ à partir de l'ensemble d'apprentissage $(x_i, z_i)_{i=1}^{n}$. On considère le score $f\left(x_j \middle| z = k\right)$ suivant qui calcule la «distance» du spectre candidat x_j à la classe k :

$$f\left(x_j\middle|z=k\right)=\frac{1}{n_k}\sum_{\{i|z_i=k\}}e^{-H\left(x_j\middle||x_i\right)}, \tag{2.2}$$

avec $H\left(p||q\right)$ l'entropie croisée entre p et q : $H\left(p||q\right)=-\sum_{j=1}^{N}p_j\log\left(q_j\right)$.

La probabilité de la classe k est ensuite calculée en normalisant ces scores par la formule de Bayes :

$$p\left(z=k\middle|x_j\right)=\frac{f\left(x_j\middle|z=k\right)p\left(z=k\right)}{\sum_{h=1}^{g}f\left(x_j\middle|z=h\right)p\left(z=h\right)} \tag{2.3}$$

avec $p\left(z=k\right)=n_k/n$. Sous hypothèse d'indépendance conditionnelle des évènements $z=k|x_j$ $(m=1,...,M)$, les probabilités conditionnelles sont ensuite agrégées temporellement sur les $m-n$ sons partiels :

$$p\left(z=k\middle|x_{n+1},...,x_m\right)=\prod_{j=n+1}^{m}p\left(z=k\middle|x_j\right). \tag{2.4}$$

La règle de décision est donnée par le principe du maximum a posteriori (MAP) :

$$\hat{z}=\hat{R}(x_{n+1},...,x_m)=\underset{k}{\operatorname{argmax}}\,p\left(z=k\middle|x_{n+1},...,x_m\right). \tag{2.5}$$

2.3 Réduction du temps de traitement

La complexité de l'identification est $\mathcal{O}(\sum_k n_k)$, qui est potentiellement très grande (comme on le verra dans la Partie 3). Pour réduire cette complexité, une classification hiérarchique ascendante (CAH) sur les histogrammes est réalisée. La réduction consiste à mélanger les histogrammes présents dans les clusters induits par la CAH. Une CAH nécessite une distance entre les éléments à classer et un critère de regroupement. Notre donnée de base étant l'histogramme, on considère la distance de Hellinger. Le critère de regroupement choisit pour la CAH est le critère de Ward, aussi appelé critère de minimum de variance. Le résultat de la CAH dans chaque groupe k donne un arbre de classification des différents histogrammes. On choisit le nombre de clusters à extraire de cet arbre, noté n'_k. Pour chaque cluster $\mathcal{C}_{ki}\in\{\mathcal{C}_{k1},...,\mathcal{C}_{kn'_k}\}$, on note $b_{ki}=\{j\mid x_j\in\mathcal{C}_{ki}\}$ $(i=1,...,n'_k)$. L'histogramme réduit $\tilde{x}_i$ de label z_i associé au cluster $\mathcal{C}_{ki}$ est défini comme le mélange des différents histogrammes dans ce cluster :

$$\tilde{x}_i=\frac{1}{\operatorname{card}\left(b_{ki}\right)}\sum_{j\in b_{ki}}x_j. \tag{2.6}$$

Le nouvel ensemble d'apprentissage consiste donc en les couples $(\tilde{x}_i,z_i)_{i=1}^{n'}$ avec $n'=\sum_{k=1}^{g}n'_k$. Le score associé à un nouveau spectre x_j pour la classe k se calcule en remplaçant

TABLE 1 – Taux de bonne classification en % pour les différentes méthodes et bases de sons.

Base	A-Volute	ESC-10	ESC-50
Notre méthode	99,4	97,9	96,8
Méthode paramétrique	73,6	73,5	45,5
Méthode non-paramétrique	46,6	76,0	53,2
Humain	91,8	95,7	81,3

les x_i par $\tilde{x}_i$. On peut ainsi appliquer la procédure précédente pour obtenir la règle de décision.

3 Expériences

Afin de quantifier les performances de la méthode, plusieurs expériences ont été réalisées. Trois bases de données sont considérées : la base A-Volute (constituée de 704 sons répartis en 9 classes), et les bases ESC-10 et ESC-50 (Piczak (2015b)) (constituées de 400 et 2000 sons environnementaux répartis en 10 et 50 classes respectivement). Toutes ces données audio sont rééchantillonnés à $f_e = 44,1\text{kHz}$ et centrés. La taille de fenêtre est réglée à $T = 2048$ et le décalage temporel à $D = 512$ unités temporelles. La taille de la TFD est de T et on considère $N = T/5 = 410$ composantes fréquentielles dans le spectre d'énergie. Le nombre de sons partiels du son de test est fixé à $m - n = 10$; si en pratique le nombre possible de sons partiels dépasse $m - n$, on considère plusieurs blocs de $m - n$ sons partiels et on réitère la procédure sur ces blocs. Ce nombre de sons partiels correspond à 464ms de données audio. Les ensembles d'apprentissage associés aux bases de sons sont divisés suivant le schéma v-fold pour réaliser une procédure de validation croisée, avec $v = 5$. La métrique de performance est le taux de bonne classification en validation croisée. La méthode a également été comparée à d'autres techniques de classification audio : celle de Clavel (2005) (mélange de gaussiennes et descripteurs classiques), ainsi que celle de Piczak (2015a) (réseaux de neurones convolutifs profonds utilisant un spectrogramme). Enfin, des tests d'écoutes ont été réalisés pour avoir les performances d'humains sur les bases de sons considérés. Les résultats pour les bases ESC-10 et ESC-50 étant connus (Piczak (2015b)), seule la base A-Volute a été étudiée par nos soins. Au total, 21 participants ont classé 10 sons choisis aléatoirement dans les 9 classes de la base. Une estimation grossière du score de classification a été obtenu sur cette base.

Les résultats sur le dictionnaire complet sont disponibles dans la TABLE 1. Notre méthode surpasse largement les méthodes concurrentes considérées (toujours supérieur à 95% peu importe la base). Pour un fold de la base A-Volute ($n = 70000$), la décision prend 232ms pour être calculée sur un processeur Intel®Core™i7 @2.7 GHz et 13ms sur

carte graphique NVidia TitanX. Sachant que la durée d'un son partiel est de 46,4ms, le temps de calcul sur processeur est largement trop grand. Sur carte graphique ce temps est considérablement réduit mais ce matériel est coûteux et peu répandu. On considère les résultats du dictionnaire réduit sur la base A-Volute. Pour une réduction très faible ($n'_k = n_k/2$), le taux de bonne classification est de 99,3% et le temps de calcul est de 120ms. Pour une réduction très forte ($n'_k = n_k/400$), le taux de bonne classification est de 82,8% et le temps de calcul est de 0,4ms. On remarque que la construction de ce modèle réduit permet de contrôler le compromis précision - temps de calcul.

4 Discussion et conclusion

La méthode développée dans ce papier a pour but de réaliser de la classification supervisée de signaux audio en temps réel. Elle repose sur une nouvelle approche à base de modèles de mélanges d'histogramme. Dans le but de pouvoir utiliser l'algorithme en temps réel, une étape de réduction des modèles est nécessaire, basée sur une classification hiérarchique des modèles. Les performances de la méthode sont bien supérieures aux autres méthodes de l'état de l'art considérées dans cet article (mélange de gaussiennes et deep learning). Pour le moment, cette méthode permet de faire de l'identification mono-source (c'est-à-dire une source audio active à la fois). Néanmoins par construction, on peut étendre ce procédé pour identifier plusieurs sources présentes en même temps (contexte multi-sources) en considérant des mélanges de mélanges présents dans les modèles.

Bibliographie

[1] Baelde M., Biernacki C. et Greff R. (2017), A mixture model-based real-time audio sources classification method, *The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP2017*.
[2] Clavel C., Ehrette T. et Richard G. (2005), Events Detection for an Audio-Based Surveillance System, *2005 IEEE International Conference on Multimedia and Expo*, 1306–1309.
[3] McLachlan G. et Peel D. (2000), Finite Mixture Models, *Wiley*.
[4] Peeters G., Giordano B. L., Susini P., Misdariis N. et McAdams S. (2011), The Timbre Toolbox : extracting audio descriptors from musical signals, *The Journal of the Acoustical Society of America*, 130, 5, 2902-2916.
[5] Piczak K. J. (2015a), Environmental sound classification with convolutional neural networks, *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1-6.
[6] Piczak K. J. (2015b), ESC : Dataset for Environmental Sound Classification, *ACM Press*, 1015-1018.

Pattern Recognition 92 (2019) 82–92

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Real-Time monophonic and polyphonic audio classification from power spectra

Maxime Baelde[a,b,*], Christophe Biernacki[b], Raphaël Greff[a]

[a] A-Volute, 19 rue de la Ladrié, Villeneuve d'Ascq, 59491, France
[b] Modal team, Inria, Univ. Lille, CNRS, UMR 8524 - Laboratoire Paul Painlevé, F-59000, France

ARTICLE INFO

Article history:
Received 10 July 2018
Revised 8 March 2019
Accepted 21 March 2019
Available online 21 March 2019

Keywords:
Real-time
Audio classification
Machine learning
Monophonic
Polyphonic
Generative model
Nonparametric estimation

ABSTRACT

This work addresses the recurring challenge of real-time monophonic and polyphonic audio source classification. The whole normalized power spectrum (NPS) is directly involved in the proposed process, avoiding complex and hazardous traditional feature extraction. It is also a natural candidate for polyphonic events thanks to its additive property in such cases. The classification task is performed through a nonparametric kernel-based generative modeling of the power spectrum. Advantage of this model is twofold: it is almost hypothesis free and it allows to straightforwardly obtain the *maximum a posteriori* classification rule of online signals. Moreover it makes use of the monophonic dataset to build the polyphonic one. Then, to reach the real-time target, the complexity of the method can be tuned by using a standard hierarchical clustering preprocessing of the prototypes, revealing a particularly efficient computation time and classification accuracy trade-off. The proposed method, called RARE (for Real-time Audio Recognition Engine) reveals encouraging results both in monophonic and polyphonic classification tasks on benchmark and owned datasets, including also the targeted real-time situation. In particular, this method benefits from several advantages compared to the state-of-the-art methods including a reduced training time, no feature extraction, the ability to control the computation - accuracy trade-off and no training on already mixed sounds for polyphonic classification.

1. Introduction

Audio source classification has been a challenging research subject for the past thirty years, beginning with speech recognition [1], and currently known as the vast field of *sound event detection* (SED). The latter consists in detecting and classifying audio sources present in *monophonic* (one source active at a time) and *polyphonic* (several sources at a time) audio streams. Many methodologies and algorithms were created to solve SED, and can be distributed in three main topics. The most prominent topic is automatic speech recognition (ASR) whose goal is to identify speech (in particular phonemes) in audio recording [2,3]. The next topic is music information retrieval aiming at analyzing musics and extracting relevant information such as the musical genre [4] (rock, classical, *etc.*) or the different instruments [5]. The last topic is environmental sound recognition which aims at recognizing sounds such as airplanes, dog coughs, trains, gunshots, *etc.* [6–8]. SED can be performed offline – using the whole signal – or online – audio data come on the fly as *time frames*. Online or *real-time* processing relates to two criteria [9]: *speed* and *latency*. First, speed is the time to make the decision and is related to how many time frames the system uses (past and future). Second, latency is related to the computation time. For instance, if a time frame lasts 50ms, the decision has to be made within these 50ms, otherwise the result will not be used in the process. The latter is illustrated on Fig. 1.

State-of-the-art SED methods typically involve two steps: *feature extraction* and *supervised learning*. Feature extraction – a universal stage in Machine Learning – summarizes the available information with a set of (expected) discriminant features. The usual audio features are known as *audio descriptors* [10], distributed in three groups. Temporal features use the raw audio signal (as a function of time) and consist in the energy, the autocorrelation coefficients and the zero crossing rate (*i.e.* how many times the signal crosses zero) for instance. Spectral features are extracted using the Fourier Transform (FT) and are mainly the spectral moments (centroid, spread, skewness, kurtosis). Cepstral and perceptual features are computed using the inverse FT of the log-magnitude FT on Mel-scale (for the Mel Frequency Cepstral Coefficient (MFCC)),

* Corresponding author at: A-Volute, Villeneuve d'Ascq, 59650, France.
E-mail addresses: maxime.baelde@a-volute.com (M. Baelde), christophe.biernacki@math.univ-lille1.fr (C. Biernacki), raphael.greff@a-volute.com (R. Greff).
URL: http://mbaelde.lille.inria.fr (M. Baelde)

https://doi.org/10.1016/j.patcog.2019.03.017
0031-3203/

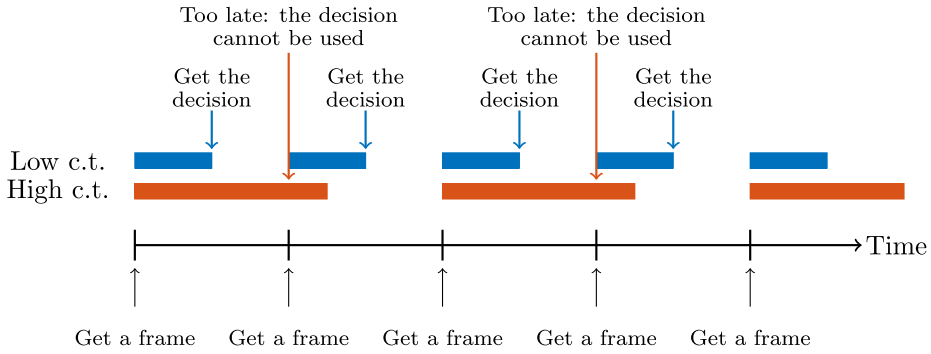


Fig. 1. Illustration of the latency of a real-time audio classification system. A real-time process gets the frames at specific time clocks (black bottom arrows). If the computation time (c.t.) is low (Low c.t., blue filled rectangles), the decision will be used by the system because it will be available before the next frame. If the computation time is high (High c.t., red filled rectangles), the decision is discarded because it cannot be used by the system. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

and a harmonic decomposition (for the fundamental frequency, the inharmonicity, *etc.*), respectively. It should be worth noting that temporal features can also be extracted by considering the temporal evolution of the spectral and cepstral features [5]. The relevance of the features depends on the context: for instance, the MFCC are good features for glass break, but not for gun shot recognition [11].

Several supervised learning methods have been applied to monophonic SED using the previous features. What is called "monophonic" in the SED field of research corresponds to a multi-class single-label classification task. First, Gaussian Mixture Models (GMM) are generative models assuming that the distribution generating the data given a class is a mixture of Gaussian distributions. The decision is computed using the *maximum a posteriori* (MAP), which is the maximum posterior probability of the classes. This modeling has been applied with MFCC for gunshot detection [12,13] and real-time voice detection in medical application [14,15]. Second, Hidden Markov Models (HMM) are used to model the temporal continuity of audio signals, alone or coupled with GMM. Bietti et al. modeled the normalized audio spectra using HMM in an online setting [16], whereas Heittola et al. fed a HMM with histograms of MFCC [17]. Third, Support Vector Machines (SVM) are binary classifiers that map the features into a high dimensional space using the kernel trick and perform the classification in this space. The SVM can be extended to multi-class classification by considering learning strategies like One-Versus-One or One-Versus-All. SVM have been used with the energy and the signal spectrogram in [18] and [19] for surveillance application, and also in [20] for meeting room sounds. Finally, Neural Networks (NN) – and their deep variants such as Deep Convolutional/Recurrent Neural Networks (DC/RNN) – have recently attracted the attention of researchers. Biondi et al. used a normalized spectrum as input for a standard NN [21] and Dadula et al. considered the MFCC [22]. Palaz et al. [23] and Piczak [6] considered directly the raw audio signal and the spectrogram, respectively, using a deep architecture (DCNN). Other authors considered also Random Forest [24] using more contextual information.

Other algorithms are used in case of polyphonic SED. What is called "polyphonic"[1] in the SED field of research corresponds to a multi-class multi-label classification task. It is often cast to a multi-class single-label classification task but the output of the system is thresholded to predict the active classes. Çakır et al. [25] developed a Convolutional Recurrent Neural Network (CRNN), taking advantages of the two structures (convolutional and recurrent). However, NN-based algorithms require a huge amount of labeled data to train the network, which is not always available (no public dataset or time-consuming data recording and labeling). Apart from NN, two main modeling algorithms are considered: PLCA (Probabilistic Latent Component Analysis) [26] and NMF (Nonnegative Matrix Factorization) [27]. Benetos et al. [28] proposed a probabilistic modeling of ERB (Equivalent Rectangular Bandwidth) spectra using PLCA coupled with HMM. NMF has been used as a task-driven modeling of MFCC spectra [29] or as coupled training of sound spectra and class annotation [30]. Heittola et al. [31] constructed a two-step method based on unsupervised source separation (with NMF) and classification of the separated sources (using a GMM-HMM).

The previous review arouses three unresolved problems. First, the methods rely on context-based (which are not always relevant) features. Second, the algorithms are not often designed for real-time processing: either they need lots of frames (low speed) or the computations are too heavy (high latency). Third, typical methods for polyphonic SED suffer from three drawbacks: the number of active sources is assumed to be known, the output of the system is thresholded, and a dataset of sound mixtures is needed. Even for general multi-label learning task, usual methods include Binary Relevance (BR, learn a classifier for each label individually: simple but does not learn correlations between labels) or Label Powerset (LP, consider multi-label output as a new single-label: complex and combinatorial) [32], which are far from optimal solutions. More advanced multi-label methods also exist such as RAKEL [33] (combination of LP classifiers) or Classifier Chains [34] (chain of BR classifiers) and their ensemble versions. To overcome the previous problems, a novel method is proposed in this paper that can perform both monophonic and polyphonic real-time SED without assuming the number of active sources to be known and by using the audio spectrum itself (and not audio descriptors) for the classification.

The method developed in this paper, called RARE (for Real-time Audio Recognition Engine) is based on a generative model of the whole normalized power spectrum (NPS), releasing the need of (sometimes perilous) feature extraction. In particular, the use of the power spectrum instead of standard magnitude spectrum is useful for the polyphonic modeling task thanks to the additive property of uncorrelated signals. Consequently, a suitable decomposition of the polyphonic spectra using monophonic ones allows to dispense with learning mixture of sounds, which is not possible with classical predictive modeling. In addition, the generative model has two advantages. Firstly, it can be considered as a very low assumption situation since it is related to a kernel density estimation using multinomial kernels (nonparametric framework).

[1] Polyphonic SED must be distinguished from polyphonic music, the latter referring to playing different melodies simultaneously.

Secondly, it allows to straightforwardly derive a time-varying MAP for the online classification. However, using the model as is, the real-time target is not reached because of the involved computational load. A model preprocessing using hierarchical clustering is thus developed to reduce this computational load, leading finally to an efficient trade-off between accuracy and computation time. The proposed method is a worthwhile extension of the one presented in our two previous conference papers [35,36], in a more formalized way and with added extensive experiments.

The contribution of the present article can be summed up by the combination of the following three points:

1. **Features**: The *use of the whole power spectrum* instead of usual features extracted from the audio signals, which is moreover essential for polyphonic event;
2. **Modeling**: A *very general generative modeling* of the power spectra designed for real-time audio classification, that uses monophonic models to build the polyphonic models;
3. **Real-time**: A *model reduction* technique based on hierarchical clustering preprocessing of the sound models.

The paper is organized as follows. The monophonic modeling is disclosed in Section 2 and is extended to polyphonic cases in Section 3. The reduction of the complexity is detailed in Section 4. The experiments to assess the performance of the method are presented in Section 5. Finally, Section 6 concludes the paper.

2. Monophonic modeling of the classes

The monophonic modeling is a mandatory step toward the polyphonic modeling, but the main goal of this work is to perform polyphonic classification. Indeed, the polyphonic modeling will be build upon the monophonic one in a straighforward manner (see Section 3).

2.1. Problem statement

The purpose of this paper is to provide a real-time sound classification method. Consider the case where the objective is to classify sounds coming from video games (case study from the company A-Volute[2]). The classification uses only the audio mix coming from the video game, *without* any additional knowledge – metadata from the game for instance. The sound is assumed to contain events coming from some *classes of sounds* – for instance a gunshot or an airplane – which have to be inferred (see Fig. 2(a)). The objective of classifying at time t a sound can be written as follows:

$$\hat{z} = \underset{z}{argmax} p\left(z | f\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right), t\right), \tag{1}$$

where $\hat{z}$ is an estimate of the label $z \in \{1, \dots, K\}$ representing the class of sound at time t (for instance, the class $z = 1$ is composed of airplane sounds, $z = 2$ is composed of gunshot sounds, *etc.*), $\mathbf{x}^{\text{time}}$ is a sound considered as a process of length T and $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$ is the observed sound in the time interval $[t - \Delta T, t]$, ΔT is the period of observation, $f\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right)$ is a function that computes features from $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$ according to the constraints and objectives described below, and $p\left(z \middle| f\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right), t\right)$ is the probability of the class z given the features extracted from the sound $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$. The class label z is assumed to be the same over the time period $[t - \Delta T, t]$ (continuity of z and one active sound at most): this is the definition of a *monophonic frame*. However the overall sound $\mathbf{x}^{\text{time}}$ can contain instances of difference classes.

[2] A software editor company specialized in 3D sound.

Several constraints are considered for real-time classification. The first is the time related to the data acquisition which constrains t to be a multiple of the *shift length* δt, that is $t \in \{0, \delta t, 2\delta t, \dots\}$. This shift length is important because the classification has to be done at every multiple of δt, therefore the speed of the recognition system has to be less than this shift length. The second constraint is related to ΔT which is the *period of observation* of the sound which has to be set according to the processing time (see Section 2.2). The problem formulation as an argmax of a distribution is suitable because it is mathematically well-posed and it is a classical framework in probabilistic modeling. From the objective in Eq. (1), $f(\cdot)$ and $p(\cdot)$ have to be chosen carefully, and are disclosed in the following sections.

Remark. In practice, all the processing are done in discrete time, due to the processing on computers. As a result, the sound $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$ will be a real vector after sampling on the computer at a rate F (see the value F in Section 5).

2.2. Feature design

The function $f(\cdot)$ is used to compute relevant features from $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$ to perform the classification. Indeed, time domain signals are not well suited for audio classification since they are not discriminant enough features: they convey the phase information and also the volume, from which we want the method to be invariant. The sound is converted into the frequency domain: more particularly the *normalized power spectrum* is considered. Therefore by using normalized power spectrum we reach the invariance property. Moreover this transformation will be particularly relevant for polyphonic spectrum since it preserves the additivity of uncorrelated signals (see Section 3). Consequently, a possible definition of $f\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right)$ can be $f_{\text{norm}} \circ f_{\text{FT}}\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right)$, where f_{FT} is the function that computes the Fourier Transform and f_{norm} is the function that normalizes the complex spectrum to get the normalized power spectrum (defined later in Eq. (5)).

However, since the classification has to be done in real-time, such a large amount of data contained in $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$ cannot be used. This is why this piece of sound is split first of all into *time frames*:

$$\left(\mathbf{x}^{\text{time}}_{1t}, \dots, \mathbf{x}^{\text{time}}_{Nt}\right) = f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right), \tag{2}$$

where $N = \lfloor (\Delta T - \Delta t)/\delta t \rfloor$ is the number of frames that can be effectively computed within the time interval $[t - \Delta T, t]$ and each frame $\mathbf{x}^{\text{time}}_{\tau t} \in \mathbb{R}^{\Delta t}$ is defined by:

$$\mathbf{x}^{\text{time}}_{\tau t} = \mathbf{x}^{\text{time}}_{[t-\Delta T+\tau\delta t, t-\Delta T+\tau\delta t+\Delta t]}, \quad \tau = 1, \dots, N. \tag{3}$$

The frame $\mathbf{x}^{\text{time}}_{\tau t}$ is thus a portion of $\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}$ of size Δt, overlapping the previous frames by a duration $\tau\delta t$ (see Fig. 2(b)). The frame length Δt is set to a small value to allow fast processing for the Fourier Transform: as a counterpart, the frame will contain less information than a large frame (like previously). As a result, a collection of several normalized power spectra is computed instead of a single large spectrum. The framed normalized power spectra are denoted by:

$$(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}) = f\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right) = f_{\text{norm}} \circ f_{\text{FT}} \circ f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right), \tag{4}$$

where f_{FT} and f_{norm} operate frame-wise. Each normalized power spectrum is computed as follows:

$$\mathbf{x}_{\tau t} = \frac{\left|\mathbf{x}^{\text{freq}}_{\tau t}\right|^2}{||\mathbf{x}^{\text{freq}}_{\tau t}||^2}, \tag{5}$$

where $\mathbf{x}^{\text{freq}}_{\tau t} \in \mathbb{C}^B$ is the Fourier Transform of $\mathbf{x}^{\text{time}}_{\tau t}$ (only $B \leq \Delta t$ frequency bins are kept), $|\cdot|$ is the element-wise modulus and $||\cdot||$ is

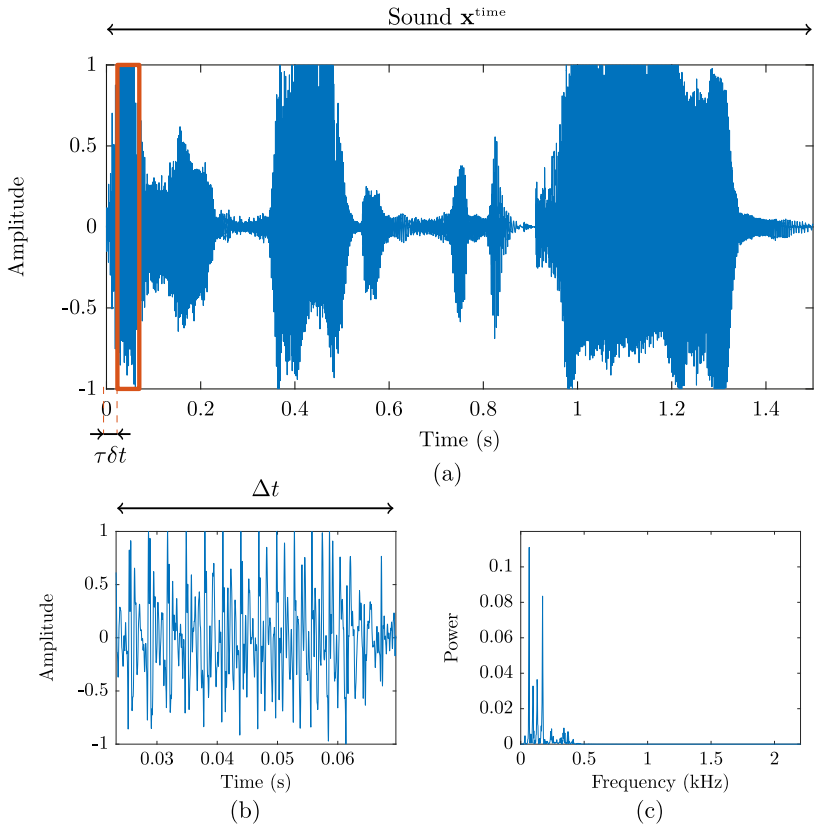


Fig. 2. (a) Example with a real sound $\mathbf{x}^{\text{time}}$ containing a voice. This sound is split into time frames $\mathbf{x}^{\text{time}}_{\tau}$ (red rectangle) of size Δt and shifted by $\tau\delta t$ (b), and each time frame is converted into the normalized power spectrum $\mathbf{x}_{\tau}$ (c). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the ℓ_2-norm (see Fig. 2(c)). The choice of Δt results in a trade-off between the computation time and the quality of the approximation: the larger is Δt the better is the approximation but the larger is the computation time.

2.3. Model design

The question now is to define the distribution $p(\cdot)$. We consider a generative model because a similar generative process will be used in the polyphonic case that simplifies the classification task. Indeed it considers only the monophonic dataset to construct the polyphonic one. This is a significant advantage compared to standard predictive modeling or usual multi-label learning. The process for generating a sequence of N spectra is detailed through the following generative model:

- **Random**: Draw a class label: $z \sim \text{Mult}_K(1, \mathbf{p})$,
- **Random**: Draw a sound of length T from class z: $\mathbf{x}^{\text{time}} \sim p_{\text{sound}}(\cdot|z)$,
- **Random**: Draw a time index: $t \sim \mathcal{U}([0, T])$,
- **Deterministic**: Compute the features: $(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}) = f\left(\mathbf{x}^{\text{time}}_{[t-\Delta T, t]}\right)$,

where $\text{Mult}_K(1, \mathbf{p})$ is a multinomial distribution over K categories and 1 draw with probabilities $\mathbf{p} = (p_k)_{k=1}^K \ \left(p_k > 0,\ \sum_{k=1}^K p_k = 1\right)$ which means that the probability of $z = k$ is p_k, and $\mathcal{U}([a, b])$ is the uniform distribution over the interval $[a, b]$. The generative modeling we adopt has the main advantage of being *hypothesis free*: the distribution that generates the class is the very general multinomial distribution and the distribution that generates the sounds is assumed to be a general distribution over real-valued processes of size T denoted by $p_{\text{sound}}(\cdot|z)$. The whole generative process is completely defined up to the distribution $p_{\text{sound}}(\cdot|z)$ which will be treated further in the conditionaling modeling.

Recall the objective in Eq. (1), we can decompose this objective using the Bayes theorem as follows:

$$p(z|\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}, t) \propto p(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}|z, t)p(z|t). \tag{6}$$

The generative process models the joint distribution $p\left(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}, t, \mathbf{x}^{\text{time}}, z\right)$, however only the conditional $p\left(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}\middle|z, t\right)$ is useful for the monophonic classification task. The full signal $\mathbf{x}^{\text{time}}$ is redundant since the frames are extracted from it so that it will not be used in the remainder. Moreover, as a consequence of the generative model, we have that $p(z|t) = p(z)$.

Several hypotheses are made to simplify the distribution. The first hypothesis is an *independence* hypothesis. Assuming that the shift length δt is large enough, two consecutive frames can be considered independent. In practice for $\delta t = \Delta t/2$ or $\delta t = \Delta t/4$ (our future chosen values) there is approximately independence between two consecutive frames. The distribution of a sequence of spectra can thus be approximated using the following conditional independence assumption:

$$p\left(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}\middle|z, t\right) = \prod_{\tau=1}^{N} p\left(\mathbf{x}_{\tau t}\middle|z, t\right). \tag{7}$$

Eq. (7) can be viewed either as an aggregation (as in [35]) or as an independence assumption. It should worth noting that this assumption can be challenging since in practice acoustic signals are temporally correlated: however for computational purpose this hypothesis is quite convenient and used in some references like [12]. The second hypothesis is an hypothesis of *stationarity in time*, so that the distribution of spectrum $\mathbf{x}_{\tau t}$ given the class z does not depend on the time t:

$$p\left(\mathbf{x}_{\tau t}\middle|z,t\right)=p\left(\mathbf{x}_{\tau t'}\middle|z,t'\right). \tag{8}$$

The third hypothesis is an hypothesis of *stationarity over the frames*, so that the distribution of a spectrum does not depend on the shift index τ (since the class is the same in the time interval $[t-\Delta T,t]$):

$$p\left(\mathbf{x}_{\tau t}\middle|z\right)=p\left(\mathbf{x}_{\tau' t}\middle|z\right). \tag{9}$$

Given Eqs. (8) and (9), the indexes τ, t can be removed so that $\mathbf{x}_{\tau t}=\mathbf{x}$, and finally the distribution can be written:

$$p\left(\mathbf{x}_{\tau t}\middle|z,t\right)=p\left(\mathbf{x}\middle|z\right). \tag{10}$$

2.4. Estimation of the model

We suppose that a learning set $\left(\mathbf{x}_{(i)},z_{(i)}\right)_{i=1}^{n}$ is available using the generative process introduced in the previous section. The conditional distribution of Eq. (10) is estimated using a *kernel density estimation* of the form:

$$\widehat{p}(\mathbf{x}|z)=\frac{1}{n_z}\sum_{\substack{i\\ z_{(i)}=z}}K\left(\mathbf{x},\mathbf{x}_{(i)}\right), \tag{11}$$

where n_z is the number of samples in the class z and $K(\cdot,\mathbf{x}_{(i)})$ is a kernel that has to be chosen. From the definition of $\mathbf{x}$, several kernels could be used: the Dirichlet kernel – though it is not flexible enough for our purpose –, a mixture of Dirichlet kernel – which is more flexible but in a real-time context the involved computational load is too high – and finally a very classical Gaussian mixture models like in [35]. We have tried these three kernels (results no reported here) but the best results were obtained with the very simple multinomial kernel that we present now. The *multinomial kernel* quantizes the spectrum so that it becomes a vector of integers that sums to a quantization factor $q\in\mathbb{N}$. Indeed this kernel gathers the advantages to be easy to evaluate, and reaches an efficient computation time - accuracy trade-off as described later in Section 4, because only dot products are required to compute the distribution (the normalization constant is the same for each kernel). Consider $\mathbf{x}_{(i)}^{(q)}\in\mathbb{N}^B$ (where B is the number of frequency bins) the closest integer vector to $q\mathbf{x}_{(i)}$ which sums to q defined by:

$$\mathbf{x}_{(i)}^{(q)}=\underset{\mathbf{x}\in\mathbb{N}^B}{\operatorname{argmin}}\left|\left|q\mathbf{x}_{(i)}-\mathbf{x}\right|\right|. \tag{12}$$

We define the so-called approximated kernel by:

$$K^{(q)}\left(\cdot,\mathbf{x}_{(i)}^{(q)}\right)=\mathrm{Mult}_B\left(\cdot;q,\mathbf{p}_{(i)}^{(q)}\right), \tag{13}$$

where the parameter $\mathbf{p}_{(i)}^{(q)}$ is defined by:

$$\mathbf{p}_{(i)}^{(q)}=\frac{1}{q}\mathbf{x}_{(i)}^{(q)}. \tag{14}$$

The approximated kernel $K^{(q)}(\cdot,\mathbf{x}_{(i)}^{(q)})$ converges to $K(\cdot,\mathbf{x}_{(i)})$ as q goes to infinity (see Appendix A for a proof). For large enough q this approximation will be correct. In practice, values of q close to B are good enough (see Section 5). Finally, the probability of the classes are estimated by maximum likelihood:

$$\widehat{p}_z=\frac{n_z}{n}. \tag{15}$$

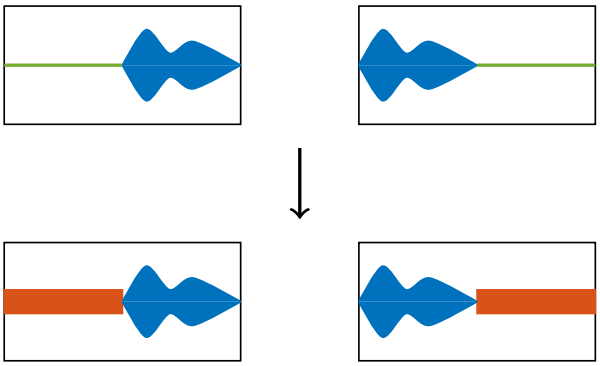


Fig. 3. We illustrate the following phenomenon. In real-time, the sound (blue filled curve) neither begins at the beginning of the frame (top left) nor ends at the ending (top right). This is why we add (bottom left and right) Gaussian white noise(red rectangle) so as to fill the "silence" (green line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Remark. In practice, the learning set is built using a slightly different generative process for practical reasons. A set of sounds already sampled from $p_{\mathrm{sound}}(\cdot|z)$ is supposed to be available. We first draw a class label $z_{(i)}$ and then a sound. The step of drawing a time index is different but *mimick* the generative process: every time index is considered and creates several time frames. The remaining of the process is the same: the time frames are transformed into the normalized power spectrum. The framing implies that in a given frame the sound neither necessarily begins at the beginning of this frame nor ends at the ending. Rather than adding silence – which contains some information – Gaussian white noise (GWN) is added – which conveys no statistical information – to fill this "blank" (see Fig. 3).

3. Polyphonic modeling of the classes

3.1. Motivation and polyphonic features

The polyphonic classification relies on the monophonic dataset, built in Section 2.4 and uses a similar framework than the monophonic one. Using a suitable decomposition of the polyphonic spectrum, the method uses only the monophonic spectra and does not have to learn mixture of sounds: it is a clear advantage of the proposal. The case for a mixture of two sources is considered but it can be straighforwardly extended to a greated number of sources: this means that a frame has two simultaneous labels, denoted by $\mathbf{z}=(z_1,z_2)\in\{1,\dots,K\}^2,\ z_1\neq z_2$. Recalling the objective in Section 2, the polyphonic decision rule is thus written as:

$$\widehat{\mathbf{z}}=\underset{\mathbf{z}}{argmax}\,p\left(\mathbf{z}|f\left(\mathbf{x}_{[t-\Delta T,t]}^{\mathrm{time}}\right),t\right), \tag{16}$$

where $\mathbf{x}_{[t-\Delta T,t]}^{\mathrm{time}}$ is the observed polyphonic sound, defined by the sum of the monophonic sounds $\mathbf{x}_{1[t-\Delta T,t]}^{\mathrm{time}}$ and $\mathbf{x}_{2[t-\Delta T,t]}^{\mathrm{time}}$ from the corresponding classes z_1 and z_2, as:

$$\mathbf{x}_{[t-\Delta T,t]}^{\mathrm{time}}=\mathbf{x}_{1[t-\Delta T,t]}^{\mathrm{time}}+\mathbf{x}_{2[t-\Delta T,t]}^{\mathrm{time}}. \tag{17}$$

The same features are used since the power spectrum keeps the additivity of the spectra – it was designed to this purpose. Then, the polyphonic feature can be expressed as a combination of the underlying monophonic features. Indeed, some calculi (see Appendix B) on the features lead to the following result:

$$f\left(\mathbf{x}_{[t-\Delta T,t]}^{\mathrm{time}}\right)=\boldsymbol{\phi}_t\cdot f\left(\mathbf{x}_{1[t-\Delta T,t]}^{\mathrm{time}}\right)+\left(\mathbf{1}-\boldsymbol{\phi}_t\right)\cdot f\left(\mathbf{x}_{2[t-\Delta T,t]}^{\mathrm{time}}\right), \tag{18}$$

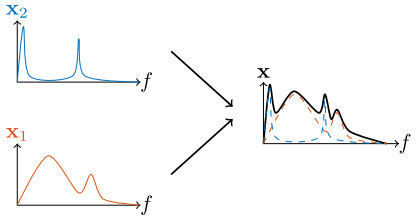


Fig. 4. Consider that two power spectra $\mathbf{x}_1$ (red, left bottom curve) and $\mathbf{x}_2$ (left top blue curve) are mixed. The resulting spectrum $\mathbf{x} = \phi\mathbf{x}_1 + (1-\phi)\mathbf{x}_2$ (thick black right curve) will depend on the relative powers of the two spectra, represented by ϕ. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where $\mathbf{x}\cdot\mathbf{y}$ is the element-wise multiplication between some vectors $\mathbf{x}$ and $\mathbf{y}$. By using normalized power spectra the modeling induces proportions $\boldsymbol{\phi}_t = (\phi_{1t}, \dots, \phi_{Nt})$ which represent the ratio between the power of one source and the power of the two sources (see Fig. 4). At a given time shift τ the proportion is defined by:

$$\phi_{\tau t} = \frac{\left|\left|\mathbf{x}^{\text{freq}}_{1\tau t}\right|\right|^2}{\left|\left|\mathbf{x}^{\text{freq}}_{1\tau t}\right|\right|^2 + \left|\left|\mathbf{x}^{\text{freq}}_{2\tau t}\right|\right|^2}, \tag{19}$$

where $\mathbf{x}^{\text{freq}}_{1\tau t}$ is defined as in the monophonic feature design Section 2.2. A detailed explanation is available in Appendix B. As already said, this framework can thus easily be extended to a mixture of more than two sources.

Remark. We assume that the time at which the sounds are observed is the same, but the individual sounds may have two different starting times: here, for the simplicity of the presentation, the underlying synchronization of the sounds is not presented but the proposed method actually takes it into account.

3.2. Model design

Based on the monophonic generative model, the following generative model of polyphonic spectra (for two classes) is defined by:

- **Random**: Draw independently without replacement two different class labels $z_1, z_2 \sim \text{Mult}_K(1, \mathbf{p})$,
- **Random**: Draw independently two different sounds: $\mathbf{x}_1^{\text{time}} \sim p_{\text{sound}}(\cdot|z_1)$ and $\mathbf{x}_2^{\text{time}} \sim p_{\text{sound}}(\cdot|z_2)$
- **Random**: Draw a time index: $t \sim \mathcal{U}([0, T])$
- **Deterministic**: Compute the features: $(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}) = \boldsymbol{\phi}_t \cdot f\left(\mathbf{x}^{\text{time}}_{1[t-\Delta T,t]}\right) + \left(\mathbf{1} - \boldsymbol{\phi}_t\right) \cdot f\left(\mathbf{x}^{\text{time}}_{2[t-\Delta T,t]}\right)$.

This polyphonic generative process models the joint distribution $p(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}, t, \mathbf{x}_1^{\text{time}}, \mathbf{x}_2^{\text{time}}, z_1, z_2)$, and again only the conditional distribution is used here. The Bayes theorem allows to write:

$$p(z_1, z_2|\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}, t) \approx p(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}|z_1, z_2, t)p(z_1, z_2|t). \tag{20}$$

From the generative model we have that $p(z_1, z_2|t) = p(z_1, z_2)$. The frame independence assumption of Eq. (7) is still valid so it leads to the following approximation:

$$p(\mathbf{x}_{1t}, \dots, \mathbf{x}_{Nt}|z_1, z_2, t) \approx \prod_{\tau=1}^{N} p(\mathbf{x}_{\tau t}|z_1, z_2, t). \tag{21}$$

Moreover the stationary hypotheses are also valid in this context (stationarity in time as in Eq. (8) and stationarity over the frames as in Eq. (9)). As $\phi_{\tau t}$ depends on $\mathbf{x}_{\tau t}$, the proportion will inherit from the stationarity hypotheses. As a consequence, the indexes are removed: $\mathbf{x}_{\tau t} = \mathbf{x}$ and $\phi_{\tau t} = \phi$, and the polyphonic conditional distribution can be expressed as:

$$p(\mathbf{x}_{\tau t}|z_1, z_2, t) = p(\mathbf{x}|z_1, z_2). \tag{22}$$

3.3. Polyphonic dataset building and estimation of the model

The main advantage of the previous modeling is that it allows to build a new dataset for polyphonic sounds using only monophonic ones: this releases the need to record and label manually polyphonic events which can be hard and time consuming. The procedure is rather simple: based on the polyphonic generative model, we first draw two class labels from which two sounds are drawn. Then each sound is split in time frames and converted into normalized power spectra, and the energy of each frequency domain frame is computed. Finally pairwise normalized spectra are computed by weighting two normalized monophonic spectra by the corresponding proportion $\phi_{(i)}$ (defined in Eq. 19), with the form:

$$\mathbf{x}_{(i)} = \phi_{(i)}\mathbf{x}_{1(i)} + \left(1 - \phi_{(i)}\right)\mathbf{x}_{2(i)}. \tag{23}$$

This procedure leads to a learning set $\left(\mathbf{x}_{(i)}, \mathbf{z}_{(i)}\right)_{i=1}^{n}$, where $\mathbf{z}_{(i)} = \left(z_{1(i)}, z_{2(i)}\right)$. The previous distribution is estimated using a *kernel density estimation* of the form:

$$\widehat{p}(\mathbf{x}|\mathbf{z}) = \frac{1}{n_{z_1} n_{z_2}} \sum_{\substack{i \\ \mathbf{z}_{(i)}=\mathbf{z}}} K\left(\mathbf{x}, \mathbf{x}_{(i)}\right). \tag{24}$$

As for the monophonic estimation, the kernel will be approximated using a multinomial kernel with the same convergence property. By construction, this method can theoretically only recover polyphonic sounds with the same proportions $\phi_{(i)}$ as in the learning set, but we will see in Section 5 that the method performs well even with random proportions between sounds.

4. Reducing the computational load

The main objective of this paper is to provide a real-time audio classification method for both monophonic and polyphonic sounds. The previous two sections defined such a method from a real-time point of view (split sounds in short frames, use a kernel density estimate with multinomial kernels fast to compute). However, the computational load for computing the distribution is currently too high to be used in practice. This is the point of this section, that is to reduce the computational load.

4.1. Evaluating the complexity of the classification task

Consider the following writing of the monophonic distribution in Eq. (11):

$$\widehat{p}(\mathbf{x}|z) \propto \sum_{\substack{i \\ z_{(i)}=z}} \exp\left(\left(\mathbf{x}^{(q)}\right)^{\top} \log\left(\mathbf{p}_{(i)}^{(q)}\right)\right), \tag{25}$$

where $\mathbf{x}^{\top}$ is the transpose $\mathbf{x}$. The complexity of the algorithm is roughly $\mathcal{O}(n)$ since it consists in computing dot products between the unknown spectrum and all the $\mathbf{p}_{(i)}^{(q)}$. Even for small datasets, the number of models can be very large (typically 100k prototypes), and therefore the computation time can be larger than the duration of a frame. As the previous equations are derived from the generative model (and cannot be changed), we can essentially reduce the complexity by reducing the number of prototypes n.

4.2. Hierarchical clustering of the monophonic models

A model reduction technique was already considered in [30] for NMF dictionaries: the authors used a k-means clustering technique and kept the centroid of the clusters as their reduced models. Their results suggested that the accuracy of the resulting system was not monotonic with the considered number of clusters. Contrary to these authors, the proposed model reduction algorithm of the

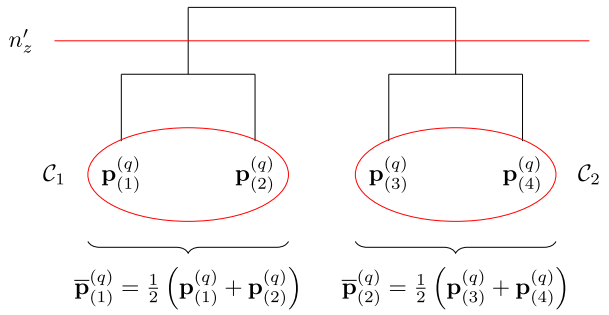


Fig. 5. Illustration of the complexity reduction using hierarchical clustering. Consider that in a given class there are 4 models parameterized by $\mathbf{p}^{(q)}_{(1)}, \dots, \mathbf{p}^{(q)}_{(4)}$. The hierarchical clustering results in a tree representing how the models are structured. Choose a number of models after reduction n' (here $n' = 2$) and "cut" the tree so as to get the clusters $\mathcal{C}_1, \mathcal{C}_2$. Finally merge the models in each cluster by taking the mean mixture of these models, to get $\overline{\mathbf{p}}^{(q)}_{(1)}$ and $\overline{\mathbf{p}}^{(q)}_{(2)}$.

present work allows to control the complexity and leads to an efficient computation time - accuracy trade-off.

The idea is to perform class-wise hierarchical clustering of the prototypes parameterized by the $\mathbf{p}^{(q)}_{(i)}$ for $i|z_{(i)} = z$, and use the resulting tree to create mixtures of these parameters according to the clusters. Hierarchical clustering requires a distance between the elements and a linkage criterion. We consider the standard Hellinger distance [37] since the elements to cluster are (discrete) probability distributions. The elements are linked using the Ward criterion [38] – a usual linkage criterion.

In class z, a hierarchical clustering is performed as described previously using a class-wise reduction factor r_z (or a global reduction r). This factor is defined as the initial number of prototypes n_z over the number of prototypes after reduction n'_z: $r_z = n_z/n'_z$. For each cluster $\mathcal{C}_{i'} \in \{\mathcal{C}_1, \dots, \mathcal{C}_{n'_z}\}$ resulting from the hierarchical clustering, $\mathcal{I}_{i'} = \left\{i : \mathbf{p}^{(q)}_{(i)} \in \mathcal{C}_{i'}\right\}$ denotes the set of the indexes such that $\mathbf{p}^{(q)}_{(i)}$ belongs to the cluster $\mathcal{C}_{i'}$ ($i' = 1, \dots, n'_z$). A mixture $\overline{\mathbf{p}}^{(q)}_{(i')}$ of the parameters $\mathbf{p}^{(q)}_{(i)}$ in cluster $\mathcal{C}_{i'}$ is defined as:

$$\overline{\mathbf{p}}^{(q)}_{(i')} = \frac{1}{\mathrm{card}(\mathcal{I}_{i'})} \sum_{i \in \mathcal{I}_{i'}} \mathbf{p}^{(q)}_{(i)}. \tag{26}$$

The label associated to $\overline{\mathbf{p}}^{(q)}_{(i')}$ is noted $z_{(i')}$. An illustration of this process is displayed in Fig. 5. It is shown in Section 5 that the reduction method improves the results up to a certain reduction value and suggests that the procedure limits the influence of overfitting.

4.3. Controlling the trade-off using a threshold procedure

A heuristic algorithm to reduce the model in a more efficient way was developed based on the previous algorithm. Instead of reducing all the classes with the same factor r_z, the classes are reduced independently by choosing the reduction factor so that the class-wise test accuracy reaches a given threshold t_{accuracy}. Consider an initial reduction factor for the classes $r^{(0)}_z$. The models are reduced using this factor and the class-wise test accuracy is computed: if it is above the threshold t_{accuracy} the procedure stops, else the factor is reduced so as to increase the accuracy. This procedure is iterated until the threshold is reached. This procedure can be used to reach a given computation time threshold instead of a class-wise test accuracy threshold.

Remark: Polyphonic modeling reduction. Experiments in Section 5 show that a very reduced model can be used to perform monophonic classification without loosing too much accuracy. Therefore polyphonic classification will be performed using the reduced models parameterized by the $\overline{\mathbf{p}}^{(q)}_{(i')}$ described in Section 4.2. The goal is to construct a new learning set containing relevant mixtures models. The method consists in computing pairwise mixture models from the different classes and reduces this collection of models using a hierarchical clustering like in Section 4.2: the reduction factor is set so that the computation time does not exceed the objective.

Remark: We emphasize on the hierarchical clustering as a means to reduce the computation load, however when this particular clustering is not suited (for large datasets for instance), a more usual clustering algorithm such as k-means can be used (as we will see in the experiments).

Table 1
Summary of the datasets used for the experiments. The polyphonic complexity is the maximum number of classes mixed simultaneously: when this complexity equals 2 that means there are at most 2 different classes are mixed simultaneously.

Dataset name	Type	Number of classes	Polyphonic complexity
A-Volute	Monophonic	5	1
ESC-10	Monophonic	10	1
A-Volute	Polyphonic	5	2
Battlefield	Polyphonic	3	3
TUT-SED	Polyphonic	6	4

5. Numerical experiments on real-world datasets

5.1. Databases and baseline systems

Databases. Two datasets are considered for monophonic classification. The first is the A-Volute dataset, composed of 704 video game sounds divided into 5 classes (alarm, detonation, step, vehicle, voice). The second is the ESC-10 dataset [39], composed of 400 sounds divided into 10 classes. For polyphonic classification, the mixtures from the A-Volute dataset (created using the protocole of Section 3.3) and audio recordings from the video game Battlefield 1 are considered, the latter containing events of 3 different classes (detonation, step and voice), alone and by mixture of 2 and 3 classes. The TUT-SED 2017 [40] dataset is also considered, which contains real-life recordings in a street context. A summary of the datasets is available in Table 1.

Parameters tuning. All the sounds are resampled to $F = 44.1$kHz and centered. The considered frame size is $\Delta t = 2048$ samples (46.4ms) and the time shift is fixed to $\delta t = 512$ samples (11.6ms). Other values of Δt were tested to select the best (the results are not reported). The number of frequency bins $B = \Delta t/2 + 1$ (the highest half of the spectrum is discarded according to the Nyquist theorem). The quantization of the spectra is set to $q = B$: in practice, this seems to be an optimal number, but values above B are also good. The learning sets are divided using the v-fold scheme to perform cross-validation (with $v = 5$): precisely the split is performed on the sounds. Frame sequences length is set to $N = 40$: this corresponds to approximately 500ms of audio data to make the decision.

Evaluation metrics. For monophonic classification, the evaluation metric is the classification accuracy in cross-validation, that is the number of correctly classified frames over the total number of frames. For polyphonic classification, the evaluation metric is the segment-based error rate (E.R.) and segment-based F1 score (F1) (see [41] for a detailed explanation of these metrics): a large F1 and a low E.R. mean that the system has very good performances. The F1 score is an extension of the accuracy since in a monophonic setting the F1 score becomes the accuracy (and the error rate becomes the misclassification rate).

Table 2
Monophonic classification task ($\Delta t = 2048$). Summary of the results for the A-Volute dataset in term of accuracy, training time (in seconds) and testing time per frame (in milliseconds) for different methods. RARE stands for our proposed method and NPS is the normalized power spectrum.

Method	Features	Param.	Accuracy (%)	Training t.	Testing t.
RARE	NPS	-	82.5 (6.0)	7.6×10^1	2.4×10^2
RARE	NPS	$r = 10$	83.1 (5.6)	1.0×10^3	2.5×10^1
		$r = 100$	79.9 (5.7)		2.7×10^0
GMM [12]	MFCC	20 comp.	86.6 (8.5)	1.3×10^2	3.0×10^0
		30 comp.	84.5 (7.1)	1.9×10^2	4.1×10^0
GMM [12]	NPS	20 comp.	77.2 (7.3)	7.3×10^2	2.8×10^3
		30 comp.	88.0 (7.5)	1.3×10^2	4.0×10^3
SVM	NPS	-	36.9 (9.2)	1.2×10^4	2.9×10^1
DCNN [6]	MFCC	-	61.6 (16.0)	1.4×10^3	3.1×10^0
CRNN [25]	MFCC	-	19.2 (1.6)	4.0×10^2	2.9×10^1
Human	-	-	91.7	-	-

Table 3
Monophonic classification task ($\Delta t = 2048$). Summary of the results for the ESC-10 dataset in term of accuracy, training time (in seconds) and testing time per frame (in milliseconds) for different methods. RARE stands for our proposed method and NPS is the normalized power spectrum.

Method	Features	Param.	Accuracy (%)	Training t.	Testing t.
RARE	NPS	-	64.7 (3.2)	8.2×10^1	3.7×10^2
RARE	NPS	$r = 10$	67.5 (4.4)	7.9×10^2	3.7×10^1
		$r = 100$	67.1 (5.1)		4.0×10^0
GMM [12]	MFCC	20 comp.	78.1 (4.0)	2.2×10^2	3.0×10^0
		30 comp.	78.3 (3.5)	3.0×10^2	3.5×10^0
GMM [12]	NPS	20 comp.	56.7 (3.5)	1.2×10^3	5.7×10^3
		30 comp.	56.1 (4.2)	2.0×10^3	8.0×10^3
SVM	NPS	-	46.3 (4.4)	2.2×10^4	4.7×10^1
DCNN [6]	MFCC	-	40.7 (6.0)	2.1×10^3	1.3×10^0
CRNN [25]	MFCC	-	10.0 (0.0)	6.0×10^2	2.6×10^1
Human	-	-	95.7	-	-

Monophonic baseline systems. The proposed method, denoted by RARE, for monophonic classification is compared with several baseline systems. The first is a GMM with 20 and 30 components per classes using 20 MFCC and their first and second derivatives, inspired by Clavel et al. [12]. The second is also a GMM with 20 and 30 components per classes but using the proposed normalized power spectrum (NPS) as a feature. The third is a multi-class Support Vector Machine trained using error-correcting ouput code with one-*versus*-one coding design. The fourth is a Deep Convolutional Neural Network (DCNN) using log-mel spectrograms inspired by Piczak [6], trained on 50 epochs. Human listening results were gathered for the two monophonic datasets: the results for ESC-10 are available in [39]. For the A-Volute dataset an experiment was carried out where 27 subjects classified 50 0.5s-sounds randomly chosen in the 5 classes.

Polyphonic baseline systems. The polyphonic classification method RARE was also compared with a neural network based method, called CRNN (Convolutional Recurrent Neural Network) inspired by Çakır et al. [25], trained on 100 epochs. It uses log-mel spectrogram to feed a neural network that consists in convolutional layers (for feature extraction) followed by recurrent layers (for temporal detection) and ended by a dense layer that performs classification. As this method usually uses dataset with already mixed sounds, we created artificial mixtures on the A-Volute dataset to train the network. Once the network is trained, a threshold th is used to predict the labels based on the output scores: the default value is 0.5 but we also tested other values (from 0.1 to 0.5).

Remark: polyphonic dataset creation. The polyphonic training data was created by considering a reduced monophonic training dataset. Indeed, the number of possible mixtures from the original dataset was too large to fit a standard computer memory, so that the mixtures are created using a reduced dataset. We define r_{mono} the reduction factor of the monophonic prototypes and r_{poly} the reduction factor of the polyphonic prototypes. For $r_{\text{mono}} = 100$, the monophonic dataset is reduced by a factor 100 and mixtures were created using this reduced dataset. Then the mixtures were classified using the reduced monophonic prototypes and the polyphonic ones. The CRNN has to be trained using a dataset containing mixed sounds, this is why artificial mixtures were created on the A-Volute dataset. The parameter N_{mixt} controls the number of frames from each class mixed together: if $N_{\text{mixt}} = 200$, there were 200 frames from the first class mixed with 200 frames from the second class.

5.2. Results on the monophonic classification

Proposed method without reduction. Tables 2 and 3 summarize the results for the A-Volute and ESC-10 datasets respectively for: the different methods (RARE, GMM, SVM and DCNN), the different features (NPS or MFCC) and the corresponding tuning parameter (if any). The results are the accuracy (in %), the training time (in seconds) and the testing time (in milliseconds). In the case of the GMM, the hyperparameter to tune is the number of components in the mixture. The reduction factor r in the proposed method is not a truly hyperparameter since it controls the trade-off between accuracy and the testing time (see the next paragraph). In term of accuracy, the best method is the GMM with a relatively few number of components but the proposed method is the second one and achieves good results compared to the other methods, in particular the DCNN and the SVM. We can see that on the A-Volute database, the GMM with NPS performs quite well compared to the GMM with MFCC, which indicates that the NPS is an effective space to describe sounds. The results on the ESC-10 dataset for the proposed method decrease because one class (over the 10) is not well recognized, but overall the results are similar than for the A-Volute dataset.

Proposed method with reduction. The model reduction algorithm seems to be very efficient because the number of prototypes can be reduced by a factor 100 without loosing too much accuracy, but it decreases the testing time dramatically (see last column of Tables 2 and 3). For a reduction factor of $r = 10$ the proposed method can be used in real-time because the testing time is lower than 50ms (which corresponds to the frame length). Moreover, for reduction factors above to 10, the results decrease: this may mean that the method overfits when there is no reduction and this reduction tends to improve the results. However for a large reduction factor there are less prototypes to compute the classification rule so that the results decrease.

Pros and Cons. The proposed method has several advantages compared to the state-of-the-art methods. One advantage is that there is no feature extraction. The monophonic proposed method is not the best method in term of accuracy: a fine tuned GMM can have better results. However, the GMM needs to have a number of components set beforehand or optimized using a model selection criterion (such as Bayesian Information Criterion or the classification accuracy), which increases the training time. The DCNN has a complex network architecture which is hard to fine tune and which is time consuming. The proposed method has this second advantage that the training time is small compared to the other methods. Finally, with the model preprocessing, the proposed method can be used in real-time. To summarize, the proposed method is the second best model (just before the fine tune GMM) and has the ability to control the trade-off between computation time and accuracy.

Table 4
Polyphonic classification task ($\Delta t = 2048$). Summary of the results for the A-Volute dataset in term of F1 score (F1), error rate (E.R.), training time (in seconds) and testing time per frame (in milliseconds) for different methods. r_{mono} is the reduction factor of the monophonic prototypes and r_{poly} is the reduction factor of the polyphonic prototypes. th is the threshold of the CRNN output. A large F1 and low E.R. means good performances. RARE stands for our proposed method.

Method	Tuning param.		F1	E.R.	Training t.	Testing t.
RARE	$r_{\text{mono}} = 400$	-	69.4 (3.2)	46.2 (3.2)	7.7×10^1	6.4×10^0
RARE	$r_{\text{mono}} = 100$	-	72.3 (4.8)	43.1 (5.4)	1.9×10^4	7.9×10^2
		$r_{\text{poly}} = 20$	71.4 (3.8)	41.3 (4.5)		4.2×10^1
RARE	$r_{\text{mono}} = 50$	-	74.5 (4.7)	40.6 (5.1)	7.7×10^1	3.0×10^3
CRNN [25]	$N_{\text{mixt}} = 200$	$th = 0.5$	39.8 (26.5)	70.3 (20.0)	4.8×10^3	2.8×10^1
		$th = 0.2$	56.4 (3.8)	114.4 (6.0)		
		$th = 0.1$	55.9 (2.1)	141.7 (15.0)		
CRNN [25]	$N_{\text{mixt}} = 300$	th = 0.5	53.1 (0.6)	60.3 (0.6)	1.3×10^4	3.1×10^1
		$th = 0.2$	56.2 (1.9)	141.1 (14.4)		
		$th = 0.1$	57.1 (0.1)	149.1 (1.9)		

Table 5
Polyphonic classification task ($\Delta t = 2048$). Summary of the results for the Battlefield dataset in term of F1 score (F1), error rate (E.R.), training time (in seconds) and testing time per frame (in milliseconds) for different methods. r_{poly} is the reduction factor of the polyphonic prototypes. th is the threshold of the CRNN output. A large F1 and low E.R. means good performances. RARE stands for our proposed method.

Method	Tuning param.	F1	E.R.	Training t.	Testing t.
RARE	-	66.0 (4.3)	44.5 (6.3)	5.1×10^1	9.4×10^1
RARE	$r_{\text{poly}} = 10$	69.2 (2.8)	40.9 (2.6)	1.2×10^2	1.0×10^1
	$r_{\text{poly}} = 50$	67.9 (1.8)	42.3 (2.2)		2.1×10^0
CRNN [25]	$th = 0.5$	61.8 (3.4)	54.4 (4.4)	1.8×10^2	2.6×10^1
	$th = 0.3$	63.5 (4.0)	77.9 (17.3)		
	$th = 0.2$	65.2 (2.8)	85.1 (7.0)		
	$th = 0.1$	56.1 (1.5)	156.7 (9.4)		

Table 6
Polyphonic classification task ($\Delta t = 2048$). Summary of the results for the TUT-SED dataset in term of F1 score (F1), error rate (E.R.), training time (in seconds) and testing time per frame (in milliseconds) for different methods. r_{poly} is the reduction factor of the polyphonic prototypes. th is the threshold of the CRNN output. A large F1 and low E.R. means good performances. RARE stands for our proposed method.

Method	Tuning param.	F1	E.R.	Training t.	Testing t.
RARE	-	30.1 (4.1)	85.4 (9.8)	5.4×10^2	1.3×10^3
RARE	$r_{\text{poly}} = 100$	40.2 (9.8)	60.1 (9.7)	8.8×10^2	1.6×10^1
	$r_{\text{poly}} = 1000$	47.9 (11.9)	59.1 (11.3)	8.8×10^2	2.5×10^0
CRNN [25]	$th = 0.5$	0.0 (0.0)	100.0 (0.0)	2.3×10^3	2.9×10^1
	$th = 0.4$	7.5 (15.0)	100.0 (0.0)		
	$th = 0.3$	25.2 (16.9)	99.1 (2.5)		
	$th = 0.2$	39.1 (7.7)	137.4 (57.9)		
	$th = 0.1$	34.5 (5.2)	308.4 (122.1)		

5.3. Results on the polyphonic classification

Proposed method without reduction. Table 4 summarizes the results on the A-Volute dataset for the polyphonic classification task in the same way as in the monophonic classification task. This table shows that the proposed method outperforms the CRNN in term of scores and testing time. When the reduction r_{mono} decreases, the scores are better since more prototypes are used to construct the polyphonic dataset, but the testing time increases dramatically. We also see that the threshold *th* of the CRNN controls a trade-off between the F1 and E.R.: the default value 0.5 seems to be the optimal trade-off. However the results are far worst than our method.

Tables 5 and 6 summarize the results for the Battlefield and TUT-SED dataset like previously. The main difference between these datasets and the A-Volute dataset is that mixtures are already present in the datasets. This is why the proposed method uses only a reduction factor r_{poly} for the polyphonic classes. For the Battlefield dataset, the proposed method performs better both in term of scores (F1 and error rate) and times (training and testing). For the TUT-SED dataset, with no reduction, the proposed method does not reach the real-time objective and performs badly: since this database contains 6 classes with at most 4 classes mixed together at the same time, there are equivalently around 50 classes which are effectively present so that a completely random decision would have an accuracy of approximately 2%. It is then now interesting to express the following reduction technique to reach the real-time target.

Proposed method with reduction. For a smaller r_{mono}, the proposed method for the A-Volute dataset must use a polyphonic reduction step to work in real-time. Like the monophonic reduction, the polyphonic reduction increases the results so that without reduction there may be overfitting of the proposed method. The Battlefield dataset shows a similar behaviour: better scores for a reduction of a factor 10 and then a decrease for a large reduction factor. The TUT-SED dataset was reduced using a k-means algorithm since the hierarchical clustering was not able to reduce a large amount of prototypes (more than 100k): the results are better with the reduction, which shows again that the reduction brings accuracy benefits in our case.

Pros and cons. All the previous methods used for monophonic classification are not designed to work on polyphonic classification using a monophonic dataset, contrary to our proposed method (RARE), which is a major advantage. Indeed the proposed method does not have to learn the mixtures of sounds but only the individual sounds. Moreover, the reduction of the complexity has a regularizing effect which is very effective on the TUT-SED database for instance. Even with multiple reduction of the complexity, the training time of our proposed method is still lower than the CRNN, which is trained using iterative algorithms such as stochastic gradient descent.

6. Conclusion

This work dealt with a new method for monophonic and polyphonic real-time audio sources classification. The method used the whole power spectrum instead of predefined audio descriptors, which is also useful for polyphonic events thanks to the additivity of uncorrelated spectra. The classification was based on a generative model of power spectra, which has the main advantage of being hypothesis free and allows to derive a temporal MAP to make the decision. Contrary to other methods like neural networks, this technique modeled both monophonic and polyphonic sources in a single framework.

As shown in the experiments, the polyphonic classification performed quite well on owned and benchmark datasets and outperforms the CRNN. Thanks to the reduction of the complexity, the method has a low computation time and can be used in real-time. This polyphonic classsification is built on monophonic sounds and does not have to learn from already mixed sounds, which is a major advantage compared to other methods like neural networks or Gaussian mixture models for instance. As mentionned in the experiments, the reduction of the complexity has a regularizing effect in addition to an efficient computation time - accuracy trade-off.

Since the NPS feature was essentially designed to handle the polyphonic classification, it is not surprising that the monophonic classification does not fully compete with usual monophonic classification algorithms such as a fine tuned GMM. Moreover, there may be overfitting of our method because of the nonparametric estimation using kernels. As the proposed method is flexible regarding the choice of the kernel, we also tested the Dirichlet kernel and the bGMM kernel (which is the GMM for binned data [35]). The Dirichlet kernel is constructed by considering that the NPS is the mode of the Dirichlet kernel. This kernel does not improve the results as the accuracy is 48.30% for a real-time application (after a reduction by 20). The bGMM is learned using the quantized NPS and by selecting the number of components using ICL [42] (Integrated Classification Likelihood). This kernel does not improve the results either since the accuracy is 82.7%. A future area of research would be to change the estimation of the condition distribution of Eqs. (11) and (24), using a semiparametric estimation for instance. However it is worth noting that the considered nonparametric estimation using multinomial kernels allows to derive an efficient computation - accuracy trade-off.

Appendix A. Point-wise convergence of the approximated kernel

Consider $\mathbf{X}_{(i)} \in \mathbb{R}^B$ a random vector that sums to 1 related to the spectrum $\mathbf{x}_{(i)}$ in the learning set. $\mathbf{X}_{(i)}^{(q)}$ is defined as the closest integer random vector to $q\mathbf{X}_{(i)}$ by:

$$\mathbf{X}_{(i)}^{(q)} = \underset{\mathbf{X}\in\mathbb{N}^B}{\operatorname{argmin}} \left|\left| q\mathbf{X}_{(i)} - \mathbf{X} \right|\right|. \tag{A.1}$$

We have that $\frac{1}{q}\mathbf{X}_{(i)}^{(q)}$ converges in distribution to $\mathbf{X}_{(i)}$ as q goes to infinity. As a result the kernel defined by:

$$K^{(q)}\left(\cdot, \mathbf{X}_{(i)}^{(q)}\right) = \mathrm{Mult}_B\left(\cdot; q, \mathbf{p}_{(i)}^{(q)}\right), \tag{A.2}$$

with parameter $\mathbf{p}_{(i)}^{(q)} = \frac{1}{q}\mathbf{X}_{(i)}^{(q)}$ converges to the original kernel $K(\cdot, \mathbf{X}_{(i)})$.

Appendix B. Derivation of the polyphonic spectrum decomposition

Eq. (18) is derived using the following arguments. A polyphonic sound is the sum in the time domain of several sound sources:

$$\mathbf{x}^{\text{time}}_{[t-\Delta T,t]} = \mathbf{x}^{\text{time}}_{1[t-\Delta T,t]} + \mathbf{x}^{\text{time}}_{2[t-\Delta T,t]}. \tag{B.1}$$

The framing operation is linear so that:

$$f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{time}}_{[t-\Delta T,t]}\right) = f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{time}}_{1[t-\Delta T,t]}\right) + f^{(\Delta t)}_{\text{frame}}\left(\mathbf{x}^{\text{time}}_{2[t-\Delta T,t]}\right). \tag{B.2}$$

For a given frame:

$$\mathbf{x}^{\text{time}}_{\tau t} = \mathbf{x}^{\text{time}}_{1\tau t} + \mathbf{x}^{\text{time}}_{2\tau t}. \tag{B.3}$$

By the linearity of the Fourier transform, if two signals are summed in the time domain they will be summed in the frequency domain. Denoting by $\mathbf{x}^{\text{freq}}_{1\tau t}$ and $\mathbf{x}^{\text{freq}}_{2\tau t}$ the complex spectra, the sum of these spectra $\mathbf{x}^{\text{freq}}_{\tau t}$ is:

$$\mathbf{x}^{\text{freq}}_{\tau t} = \mathbf{x}^{\text{freq}}_{1\tau t} + \mathbf{x}^{\text{freq}}_{2\tau t}. \tag{B.4}$$

The modeling disclosed in Section 2.4 requires to deal with a normalized version composed of the elements $|\mathbf{x}^{\text{freq}}_{\tau t}|^2$ as in Eq. (5). Two sources from different classes are assumed to be *uncorrelated signals* (a common assumption in signal separation [43]), meaning that the power spectrum of the sum is approximately the sum of the power spectra:

$$\begin{aligned}\left|\mathbf{x}^{\text{freq}}_{\tau t}\right|^2 &= \left|\mathbf{x}^{\text{freq}}_{1\tau t} + \mathbf{x}^{\text{freq}}_{2\tau t}\right|^2 \\ &\approx \left|\mathbf{x}^{\text{freq}}_{1\tau t}\right|^2 + \left|\mathbf{x}^{\text{freq}}_{2\tau t}\right|^2.\end{aligned} \tag{B.5}$$

The normalized power spectrum associated to $\mathbf{x}^{\text{freq}}_{\tau t}$ is $\mathbf{x}_{\tau t}$:

$$\mathbf{x}_{\tau t} = \frac{\left|\mathbf{x}^{\text{freq}}_{1\tau t}\right|^2 + \left|\mathbf{x}^{\text{freq}}_{2\tau t}\right|^2}{\left|\left|\mathbf{x}^{\text{freq}}_{1\tau t}\right|\right|^2 + \left|\left|\mathbf{x}^{\text{freq}}_{2\tau t}\right|\right|^2}. \tag{B.6}$$

Define $P_{1\tau t} = \left|\left|\mathbf{x}^{\text{freq}}_{1\tau t}\right|\right|^2$ and $P_{2\tau t} = \left|\left|\mathbf{x}^{\text{freq}}_{2\tau t}\right|\right|^2$ the powers of the two sources. Some calculi lead to the following result:

$$\mathbf{x}_{\tau t} = \mathbf{x}_{1\tau t}\frac{P_{1\tau t}}{P_{1\tau t} + P_{2\tau t}} + \mathbf{x}_{2\tau t}\frac{P_{2\tau t}}{P_{1\tau t} + P_{2\tau t}}, \tag{B.7}$$

where $\mathbf{x}_{1\tau t}$ and $\mathbf{x}_{2\tau t}$ are defined as in Eq. (5). Define the proportion $\phi_{\tau t}$ as:

$$\phi_{\tau t} = \frac{P_{1\tau t}}{P_{1\tau t} + P_{2\tau t}}. \tag{B.8}$$

The previous result becomes:

$$\mathbf{x}_{\tau t} = \phi_{\tau t}\mathbf{x}_{1\tau t} + (1 - \phi_{\tau t})\mathbf{x}_{2\tau t}. \tag{B.9}$$

References

[1] L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[2] Y. Qian, M. Bi, T. Tan, K. Yu, Very deep convolutional neural networks for noise robust speech recognition, IEEE/ACM Trans. Audio Speech Lang. Process. 24 (12) (2016) 2263–2276.
[3] X. Liu, J. Geng, H. Ling, Y.-m. Cheung, Attention guided deep audio-face fusion for efficient speaker naming, Pattern Recognit. 88 (2019) 557–568.

[4] G. Kour, N. Mehan, Music genre classification using MFCC, SVM and BPNN, Int. J. Comput. Appl. 112 (6) (2015) 12–14.
[5] C. Joder, S. Essid, G. Richard, Temporal integration for audio classification with application to musical instrument classification, IEEE Trans. Audio Speech Lang. Process. 17 (1) (2009) 174–186.
[6] K.J. Piczak, Environmental sound classification with convolutional neural networks, in: 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), 2015, pp. 1–6.
[7] P. Hu, W. Liu, W. Jiang, Z. Yang, Latent topic model for audio retrieval, Pattern Recognit. 47 (3) (2014) 1138–1143.
[8] J. Beltrn, E. Chvez, J. Favela, Scalable identification of mixed environmental sounds, recorded from heterogeneous sources, Pattern Recognit. Lett. 68 (2015) 153–160.
[9] J. Flocon-Cholet, Classification Audio Sous Contrainte de faible Latence, Université de Rennes 1, 2016 Ph.D. thesis.
[10] G. Peeters, B.L. Giordano, P. Susini, N. Misdariis, S. McAdams, The timbre toolbox: extracting audio descriptors from musical signals, J. Acoust. Soc. Am. 130 (5) (2011) 2902–2916.
[11] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, et al., DCASE 2017 Challenge setup: Tasks, datasets and baseline system, in: DCASE 2017 - Workshop on Detectionand Classification of Acoustic Scenes and Events, Munich, Germany, 2017.
[12] C. Clavel, T. Ehrette, G. Richard, Events detection for an audio-b surveillance system, in: 2005 IEEE International Conference on Multimedia and Expo, 2005, pp. 1306–1309.
[13] R. Radhakrishnan, A. Divakaran, A. Smaragdis, Audio analysis for surveillance applications, in: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005, pp. 158–161.
[14] D. Istrate, M. Binet, S. Cheng, Real time sound analysis for medical remote monitoring, in: 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2008, pp. 4640–4643.
[15] J.-H. Choi, J.-H. Chang, On using acoustic environment classification for statistical model-based speech enhancement, Speech Commun. 54 (3) (2012) 477–490.
[16] A. Bietti, F. Bach, A. Cont, An online em algorithm in hidden (semi-) Markov models for audio segmentation and clustering, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1881–1885.
[17] T. Heittola, A. Mesaros, A. Eronen, T. Virtanen, Audio context recognition using audio event histograms, in: 18th European Signal Processing Conference, 2010, pp. 1272–1276.
[18] S. Lecomte, R. Lengell, C. Richard, F. Capman, B. Ravera, Abnormal events detection using unsupervised One-Class SVM - Application to audio surveillance and evaluation -, in: 2011 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS), 2011, pp. 124–129.
[19] S. Sameh, L. Zied, On the use of time-Frequency reassignment and SVM-Based classifier for audio surveillance applications, Int. J. Image, Graph. Signal Process. 6 (12) (2014) 17–25.
[20] A. Temko, C. Nadeu, Classification of acoustic events using SVM-based clustering schemes, Pattern Recognit. 39 (4) (2006) 682–694.
[21] R. Biondi, G. Dys, G. Ferone, T. Renard, M. Zysman, Low cost real time robust identification of impulsive signals, Int. J. Comput. Electr. Automat. Control Inf. Eng. 8 (9) (2014) 1653–1656.
[22] C.P. Dadula, E.P. Dadios, Neural network classification for detecting abnormal events in a public transport vehicle, in: 2015 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2015, pp. 1–6.
[23] D. Palaz, M.M. Doss, R. Collobert, Convolutional Neural Networks-based continuous speech recognition using raw speech signal, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 4295–4299.
[24] X. Xia, R. Togneri, F. Sohel, D. Huang, Random forest classification based acoustic event detection utilizing contextual-information and bottleneck features, Pattern Recognit. 81 (2018) 1–13.
[25] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, Convolutional recurrent neural networks for polyphonic sound event detection, IEEE/ACM Trans. Audio Speech Lang. Process. 25 (6) (2017) 1291–1303.
[26] P. Smaragdis, B. Raj, M. Shashanka, A probabilistic latent variable model for acoustic modeling, In Workshop on Advances in Models for Acoustic Processing at NIPS, 2006.
[27] P. Paatero, U. Tapper, Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values, in: Fourth International Conference on Statistical methods for the Environmental Sciences Environmetrics. Espoo, Spain, Vol. 5, 1994, pp. 111–126.
[28] E. Benetos, G. Lafay, M. Lagrange, M.D. Plumbley, Detection of overlapping acoustic events using a temporally-constrained probabilistic model, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 6450–6454.
[29] V. Bisot, S. Essid, G. Richard, Overlapping sound event detection with supervised nonnegative matrix factorization, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 31–35.
[30] A. Mesaros, T. Heittola, O. Dikmen, T. Virtanen, Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 151–155.
[31] T. Heittola, A. Mesaros, T. Virtanen, M. Gabbouj, Supervised model training for overlapping sound events based on unsupervised source separation, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8677–8681.
[32] R. Alazaidah, F. Thabtah, Q. Al-Radaideh, A multi-label classification approach based on correlations among labels, Int. J. Adv. Comput. Sci. Appl. 6(2).
[33] G. Tsoumakas, I. Vlahavas, Random k-labelsets: an ensemble method for multilabel classification, Mach. Learn. 4701 (2007) 406–417.
[34] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (3) (2011) 333.
[35] M. Baelde, C. Biernacki, R. Greff, A mixture model-based real-time audio sources classification method, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2427–2431.
[36] M. Baelde, C. Biernacki, R. Greff, Classification de signaux audio en temps-réel par un modèle de mélanges d'histogrammes, 49e Journées de Statistique, 2017. Avignon, France.
[37] E. Hellinger, Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen, J. Reine Angew. Math. 136 (1909) 210–271.
[38] J.H.J. Ward, Hierarchical grouping to optimize and objective function, J. Am. Stat. Assoc. 58 (301) (1963) 236–244.
[39] K.J. Piczak, ESC: Dataset for Environmental Sound Classification, in: Proceedings of the 23rd Annual ACM Conference on Multimedia, 2015 Brisbane, Australia.
[40] A. Mesaros, T. Heittola, T. Virtanen, TUT database for acoustic scene classification and sound event detection, in: 24th European Signal Processing Conference, 2016, 2016.
[41] A. Mesaros, T. Heittola, T. Virtanen, Metrics for polyphonic sound event detection, Appl. Sci. 6 (6) (2016) 162.
[42] C. Biernacki, G. Celeux, G. Govaert, Assessing a mixture model for clustering with the integrated classification likelihood, Technical Report, INRIA, 1998. RR-3521.
[43] E.M. Grais, M.U. Sen, H. Erdogan, Deep neural networks for single channel source separation, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 3734–3738.

Maxime Baelde received M.D. degree in Applied Mathematics from Université de Lille in 2015. He is now a Ph.D. student at Inria Lille and A-Volute. His research interests are focused on real-time audio source classification and separation, and generative models.

Christophe Biernacki is the Scientific Deputy at Inria Lille and Scientific Head of the MODAL team, and received his Ph.D. degree from Université de Compiègne in 1997. His research interests are focused on model-based and model-free clustering of heterogeneous data.

Raphaël Greff is the R&D Director of A-Volute and received his Ph.D. degree from Université Paris VI in 2008. His research interests are focused on spatial audio and real-time digital signal processing.

Table des matières

Modèles génératifs pour la classification et la séparation de sources sonores en temps-réel

Résumé

Cette thèse s'inscrit dans le cadre de l'entreprise A-Volute, éditrice de logiciels d'amélioration d'expérience audio. Elle propose un radar qui transpose l'information sonore multi-canale en information visuelle en temps-réel. Ce radar, bien que pertinent, manque d'intelligence car il analyse uniquement le flux audio en terme d'énergie et non en termes de sources sonores distinctes. Le but de cette thèse est de développer des algorithmes de classification et de séparation de sources sonores en temps-réel. D'une part, la classification de sources sonores a pour but d'attribuer un label (son monophonique) ou plusieurs labels (son polyphonique) à un son. La méthode développée utilise un attribut spécifique, le spectre de puissance normalisé, utile à la fois dans le cas monophonique et polyphonique de par sa propriété d'additivité des sources sonores. Cette méthode utilise un modèle génératif qui permet de dériver une règle de décision basée sur une estimation non paramétrique. Le passage en temps-réel est réalisé grâce à un pré-traitement des prototypes avec une classification hiérarchique ascendante. Les résultats sont encourageants sur différentes bases de données (propriétaire et de comparaison), que ce soit en terme de précision ou de temps de calcul, notamment dans le cas polyphonique. D'autre part, la séparation de sources consiste à estimer les sources en terme de signal dans un mélange. Deux approches de séparation ont été considérées dans la thèse. La première considère les signaux à retrouver comme des données manquantes et à les estimer via un schéma génératif et une modélisation probabiliste. L'autre approche consiste, à partir d'exemples sonores présent dans une base de données, à calculer des transformations optimales de plusieurs exemples dont la combinaison tend vers le mélange observé. Les deux propositions sont complémentaires, avec chacune des avantages et inconvénients (rapidité de calcul pour la première, interprétabilité du résultat pour la deuxième). Les résultats expérimentaux semblent prometteurs et nous permettent d'envisager des perspectives de recherches intéressantes pour chacune des propositions.

Mots clés : temps-réel, classification audio, séparation de sources, apprentissage statistique, modèles génératifs

---

Abstract

This thesis is part of the A-Volute company, an audio enhancement softwares editor. It offers a radar that translates multi-channel audio information into visual information in real-time. This radar, although relevant, lacks intelligence because it only analyses the audio stream in terms of energy and not in terms of separate sound sources. The purpose of this thesis is to develop algorithms for classifying and separating sound sources in real time. On the one hand, audio source classification aims to assign a label (e.g. voice) to a monophonic (one label) or polyphonic (several labels) sound. The developed method uses a specific feature, the normalized power spectrum, which is useful in both monophonic and polyphonic cases due to its additive properties of the sound sources. This method uses a generative model that allows to derive a decision rule based on a non-parametric estimation. The real-time constraint is achieved by pre-processing the prototypes with a hierarchical clustering. The results are encouraging on different databases (owned and benchmark), both in terms of accuracy and computation time, especially in the polyphonic case. On the other hand, source separation consists in estimating the sources in terms of signal in a mixture. Two approaches to this purpose were considered in this thesis. The first considers the signals to be found as missing data and estimates them through a generative process and probabilistic modelling. The other approach consists, from sound examples present in a database, in computing optimal transformations of several examples whose combination tends towards the observed mixture. The two proposals are complementary, each having advantages and drawbacks (computation time for the first, interpretability of the result for the second). The experimental results seem promising and allow us to consider interesting research perspectives for each of the proposals.

Keywords: real-time, audio classification, source separation, statistical learning, generative models

---

Équipe-projet MΘDAL
Inria Lille-Nord Europe – 40 avenue du Halley – 59650 Villeneuve d'Ascq – France