



RAPPORT DE STAGE INGÉNIEUR

PRÉSENTÉ PAR STÉPHANE KOMBO

Séparation de sources multicanale par machine learning

Directeur de stage :
Alexandre GUÉRIN

Co-encadrante :
Lauréline PEROTIN

Tuteur école :
Marc CASTELLA

Mission effectuée du 12 février au 10 août 2018 chez :



Remerciements

Je tiens tout d'abord à remercier Alexandre Guérin et Lauréline Perotin, pour m'avoir donné l'opportunité d'effectuer ce stage, durant lequel ils m'ont encadré. Ce stage est certainement l'une des meilleures choses qui pouvait m'arriver à la fin de mon cursus à Télécom SudParis, et je le dois particulièrement à la confiance qu'ils m'ont accordée. Je tiens également à remercier spécifiquement Srđan Kitić, avec j'ai eu la chance de travailler au même titre qu'avec Alexandre et Lauréline, bien que cet encadrement ait été moins officiel.

Alexandre, Lauréline, Srđan, par votre exigence, votre expertise, et grâce à tout le temps que vous m'avez accordé, j'ai énormément appris. J'ai pu accomplir ma mission dans des conditions favorables à mon apprentissage, et en tirer les principales informations qu'il me manquait pour décider sereinement de mon début de carrière à l'issue de l'école.

Dans le même sens, Marc Castella, en tant que tuteur école, a également beaucoup aidé pour ces choix d'orientation, en prenant le temps de répondre à mes nombreuses questions et en m'aiguillant le mieux possible tout au long de ce stage. Je vous remercie également pour cela et pour le reste.

Mon expérience au quotidien au sein de l'équipe Traitement de la Parole et du Son de Cesson-Sévigné a été des plus agréables. Pour m'avoir accueilli et intégré, j'aimerais remercier chacun des membres de l'équipe que je n'ai pas encore cité : Marc Emerit, Arnaud Lefort, Gregory Pallone, Thomas Joubaud, Noé Faure, avec une attention particulière pour notre chef d'équipe, Stanislas Zimmermann.

Enfin, je tiens à remercier ma tante et mon oncle, Francine et Théodule Oubassou, qui m'ont gracieusement logé chez eux pendant toute la durée de cette mission. Je n'aurais jamais pu arriver à Rennes dans de meilleures conditions : vous m'avez permis de vivre avec un niveau de contraintes minimal pour me consacrer à ce stage, et le bon déroulement de ce séjour est notamment dû à votre chaleureux accueil.

Résumé

Ce rapport présente la mission de stage de fin d'études que j'ai effectuée dans le cadre de la validation de mon diplôme d'Ingénieur de Télécom SudParis. Ce stage s'est déroulé dans la section Recherche et Développement d'Orange, nommée Orange Labs, à Cesson-Sévigné. J'ai intégré l'équipe Traitement de la Parole et du Son (TPS), menant des activités dont les objectifs principaux sont l'analyse, le traitement et la restitution des contenus audio spatialisés, notamment de type ambisonique.

Dans le cadre du lancement de Djingo, l'assistant personnel à commande vocale d'Orange, des activités de recherche et de développement sont menées par l'équipe TPS. Notamment, des algorithmes internes sont mis au point afin d'évaluer les versions successives des prototypes du produit. Les principaux défis techniques que doit relever un tel assistant vocal sont la *captation en champ lointain* et le *réhaussement* du signal de voix d'un locuteur cible, en présence d'un *locuteur adverse*, au sein d'un environnement *réverbérant* et *bruyant*.

Ces problématiques sont abordées par *traitement d'antenne*. Pour l'effectuer, deux étapes fondamentales sont nécessaires : il faut d'abord *localiser* les différentes sources sonores, c'est-à-dire estimer leurs *directions d'arrivées* respectives. La cartographie ainsi obtenue permet ensuite de réaliser effectivement la *séparation de sources*, ce qui implique une *formation de voie* vers les sources considérées.

L'objectif du stage est de contribuer à ces travaux, à la fois sur l'aspect localisation et sur l'aspect séparation.

Pour la localisation, un algorithme exploitant *l'intensité acoustique* a été développé par TPS avec MATLAB. Pour des raisons de portabilité et de cohérence des travaux de l'équipe, une implémentation en Python de cet algorithme était nécessaire. Ma première mission a donc été la transcription de ce code de recherche MATLAB en un programme Python clair et maintenable. Dans ce but, j'ai développé une nouvelle version du logiciel en utilisant le paradigme orienté-objet, qui implique notamment une certaine modularité.

Pour la séparation de sources, un système utilisant des *réseaux de neurones* est en cours de développement. Les scores de performances de ce système ont été soumis en article de conférence, sur la base des expérimentations les plus prometteuses en date de la soumission. Dans l'optique d'éventuellement améliorer ces résultats, ma seconde mission a été de revoir et poursuivre les travaux déjà entrepris dans ce cadre. J'ai donc optimisé le système initial, en modifiant la méthode d'entraînement du réseau de neurones et en complétant la méthode de prédiction. Ceci étant fait, j'ai fait varier les hyperparamètres et l'architecture du réseau initial pour vérifier les pistes qui n'avaient pas encore été explorées jusque là.

Ce stage de recherche et de développement m'a donné un aperçu du métier d'ingénieur de recherche, vers lequel je m'oriente. J'ai donc choisi, en cohérence avec la nature de la mission et de mes aspirations professionnelles, de mettre en avant les aspects de l'entreprise et du stage liés à la recherche.

Abstract

This report presents the content of the end-of-studies internship I carried out as part of my engineer curriculum at Télécom SudParis. This internship took place in Orange Labs, the Research and Development division of Orange, in Cesson-Sévigné. I joined the Technologies and Processing of Sound team (TPS), leading works whose main objectives are analysis, processing and restitution of spatialized audio contents, especially ambisonic contents.

In the context of the launch of Djingo, the Orange's voice-activated personal assistant, research and development activities are leaded by the TPS team. In particular, internal algorithms are developed to evaluate successive versions of the prototypes of the product. The main technical challenges faced by such a voice assistant are the *far field capture* and *enhancement* of a target speaker's voice signal, in the presence of an *interfering speaker*, in a *reverberant* and *noisy* environment.

These issues are addressed by *microphone array processing*. To do this, two fundamental steps are needed : First we have to *localize* the different sound sources by estimating their respective *directions of arrival* ; the cartography thus obtained then makes it possible to do the effective *source separation*, which implies a *beamforming* towards the localized sources.

The aim of the internship is to contribute to these works, both on the localization and separation issues.

For the source localization, a *sound intensity* based algorithm was developed by TPS with MATLAB. For reasons of portability and consistency of the team's works, a Python implementation of this algorithm was necessary. My first mission was to transcribe this MATLAB ad hoc research code into a clear and maintainable Python program. For this purpose, I have developed a new object-oriented version of the software, which implies in particular a certain scalability.

For the source separation, a *neural networks* based system is under development. The achieved performances of this system were submitted as a conference article, based on the most promising experiments achieved at the submission date. With the ideal aim of possibly improving these results, my second mission was to review and pursue the work already undertaken in this context. So I optimized the initial system, by modifying the neural network training method and by completing the prediction method. That being done, I tested the remaining unexplored improvement tracks by trying different hyperparameters settings and architecture for the network.

This research and development internship gave me an insight of a research engineer occupation, towards which I orient myself. That is why, given the nature of the mission and my career aspirations, I chose to highlight research related aspects of the company and the internship all along this report.

Table des matières

1 Présentation de l'entreprise	1
1.1 Le groupe Orange	1
1.2 Historique technologique	1
1.3 Contexte du stage dans la hiérarchie d'Orange	2
1.3.1 L'équipe Traitement de la Parole et du Son	2
2 La mission	3
2.1 Cadre et objectifs	3
2.1.1 Un Smart Speaker pour Orange	3
2.1.1.1 Qu'est-ce qu'un assistant personnel ?	3
2.1.1.2 Intérêt du stage pour Orange : Recherche et Développement pour Djingo	4
2.1.2 Cadre scientifique : prise de son en champ lointain, localisation et séparation pour la reconnaissance automatique de la parole	5
2.2 Objectifs du stage	6
2.2.1 Localisation : Réécriture du <i>Velocity Vector Module</i> (VVM) en Python	6
2.2.2 Séparation : Reprise et poursuite des travaux sur le système de Lauréline	6
3 Le <i>Velocity Vector Module</i> (VVM)	7
3.1 Format ambisonique et environnement acoustique	7
3.1.1 Représentation du champ acoustique	7
3.1.1.1 Équation des ondes	7
3.1.1.2 Résolution en coordonnées sphériques	7
3.1.2 Format B : ambisonie à l'ordre 1	8
3.1.3 Environnement acoustique	9
3.1.3.1 Modélisation de l'effet de salle (SRIR)	9
3.1.3.2 Intensité acoustique	10
3.2 Description de l'algorithme	10
3.2.1 Vecteur intensité	10
3.2.2 Histogrammes	11
3.3 Implémentation	12
3.3.1 Comparaison MATLAB / Python	12
3.3.1.1 MATLAB	12
3.3.1.2 Python	12
3.3.1.3 Quelques différences fondamentales	13
3.3.2 Architecture du code objet	13
3.3.2.1 Fonction principale : <i>loc_multisource()</i>	13
3.3.2.2 Classe <i>RawSignal</i> : signal avant discréétisation	14
3.3.2.3 Classe <i>QuantizedSignal</i> : signal après discréétisation	14
3.4 Conclusion	15
4 Séparation multicanale par <i>machine learning</i>	16
4.1 Principes et notations du problème et de la solution	16
4.1.1 Modélisation des signaux	16
4.1.2 Description du filtrage et de la formation de voie	17
4.1.2.1 Focaliseur anéchoïque ou <i>beamformer</i> pleine bande	17
4.1.2.2 Une approximation de rang 1 du filtre de Wiener GEVD	17
4.1.2.3 Solution globale proposée	18
4.1.3 Architecture du réseau	18
4.1.3.1 Couche dense	19

4.1.3.2	LSTM	19
4.1.3.3	<i>Time Distributed Wrapper</i>	20
4.1.3.4	Hyper-paramètres	20
4.1.3.5	Normalisation	20
4.1.4	Évaluation des modèles	21
4.1.4.1	Le <i>Word Error Rate</i> (WER)	21
4.1.4.2	L'erreur quadratique moyenne (MSE)	21
4.2	Réécriture du programme	23
4.2.1	Organisation générale du système	23
4.2.1.1	Entraînement et validation	23
4.2.1.2	Test	24
4.2.1.3	Évaluation du réseau :	24
4.2.2	Nouvelle implémentation : <i>Data Generators</i> et fichiers HDF5	24
4.2.3	Modèle de référence : reproduction de la configuration de Lauréline	25
4.2.3.1	Spectrogrammes de référence	25
4.2.3.2	Résultats de référence	30
4.3	Canaux d'entrée du réseau de neurones	31
4.3.1	La normalisation	31
4.3.1.1	Canal par canal	31
4.3.1.2	Unique pour tous les canaux	31
4.3.2	Entrées de type rapport signal à bruit (SNR)	33
4.3.2.1	Rapports signaux sur mélange (SMR) et focaliseurs	33
4.3.2.1.1	Une source, $SMR(t, f)$ seul en décibels :	33
4.3.2.1.2	Une source, $SMR(t, f)$ et $\hat{s}(t, f)$ en décibels :	35
4.3.2.1.3	Deux sources, $SMR(t, f)$, $IMR(t, f)$, $\hat{s}(t, f)$ et $\hat{n}(t, f)$:	35
4.3.3	Échelle des canaux d'entrée (deux sources)	37
4.3.3.1	Entrées quadratiques	37
4.3.3.2	Entrées logarithmiques (modèle de référence)	37
4.4	Optimisation de l'apprentissage	38
4.4.1	Apprentissage pour le réseau de référence	38
4.4.2	Résultats avec $n = 12$	39
4.4.3	Dropout = 0.2	40
4.5	Conclusion	42
5	Bilan et conclusions	43

Table des figures

2.1	Exemples de <i>Smart Speakers</i>	3
2.2	Exemple de constitution d'un assistant domestique	4
2.3	Djingoo	4
2.4	Antenne sphérique à 8 microphones	5
3.1	Système de coordonées sphériques céphalo-centré. Source : [13]	7
3.2	Fonctions harmoniques sphériques de degré 0 (première ligne) à 3 (dernière ligne). Source : [1]	8
3.3	Illustration schématique d'une SRIR de TR_{60} valant 0.25 seconde. Source : [8]	9
3.4	Discrétisation des directions d'arrivées détectables par le VVM	11
3.5	Comparaison des histogrammes spatiaux instantané et lissé	11
3.6	Un histogramme en 2D et ses projections en 1D	12
4.1	Cocktail party : mélange additif de locutions concurrentes en environnement bruyant	16
4.2	Le système de réhaussement proposé par Lauréline et ses encadrants de thèse. Source : [17]	19
4.3	Architecture du réseau estimant le masque	19
4.4	WER en fonction de la MSE avec plus de réseaux différents	22
4.5	Source d'intérêt brute : $s_W(t, f)$	25
4.6	Mélange des sources : $x_W(t, f)$	26
4.7	Signal d'intérêt reconstruit avec le focaliseur pleine bande correspondant : $\hat{s}(t, f) = \mathbf{b}_0^H \mathbf{x}(t, f) $	26
4.8	Signal d'intérêt reconstruit avec le masque prédit : $\tilde{s}_W(t, f) = M_S(t, f)x_W(t, f)$	27
4.9	Masque prédict par le modèle de référence : $M_S(t, f)$	27
4.10	Signal d'intérêt reconstruit avec le masque de Wiener : $\tilde{s}_{W,idéal}(t, f) = M_{S,Wiener}(t, f)x_W(t, f)$	28
4.11	Masque de Wiener : $M_{S,Wiener}(t, f)$	28
4.12	Signal d'intérêt reconstruit avec le filtre GEVD prédict : $y(t, f) = \mathbf{w}_{GEVD}(f)^H \mathbf{x}(t, f)$	29
4.13	Signal d'intérêt reconstruit avec le filtre GEVD idéal : $y_{idéal}(t, f) = \mathbf{w}_{GEVD,idéal}(f)^H \mathbf{x}(t, f)$	29
4.14	Comparaison en <i>accuracy</i> de mon réseau et de celui de Lauréline	30
4.15	Comparaison en <i>accuracy</i> de la normalisation unique à la normalisation canal par canal	32
4.16	Comparaison en <i>accuracy</i> de la normalisation en fréquences à la normalisation canal par canal	32
4.17	« Mélange » pour une source : $x_W(t, f)$	34
4.18	Focaliseur pointant vers l'unique source : $\hat{s}(t, f)$	34
4.19	SMR de l'unique source : $SMR(t, f) = \frac{\hat{s}(t, f)}{x_W(t, f)}$	35
4.20	<i>Accuracies</i> du réseau d'entrées $SMR(t, f)$, $IMR(t, f)$, $\hat{s}(t, f)$ et $\hat{n}(t, f)$ et du réseau de référence	36
4.21	SMR pour deux sources : $SMR(t, f) = \frac{\hat{s}(t, f)}{x_W(t, f)}$	36
4.22	<i>Accuracies</i> du réseau de référence avec les entrées en décibels et linéaires	37
4.23	Les séquences prédictes pour le masque se superposent et sont redondantes	38
4.24	Prédiction des 25 - n = 13 dernières trames de chaque séquence	39
4.25	Architecture du réseau estimant le masque	39
4.26	<i>Accuracies</i> du nouveau réseau avec n = 12 et du réseau de référence	40
4.27	<i>Accuracies</i> du nouveau réseau avec $dropout = 0.2$, n = 12 et du réseau de référence	40
4.28	Cas intermédiaire : n = 3	41
4.29	<i>Accuracies</i> du nouveau réseau avec $dropout = 0.2$, n = 3 et du réseau de référence	41
4.30	<i>Accuracies</i> du réseau de référence avec $dropout = 0.2$ et $dropout = 0.5$	41
4.31	Résultats des tests de performance sur la partie séparation de sources	42

Glossaire

Beamforming : Focalisation / Formation de voie (*beamformer* : « focaliseur »)

DOA : *Direction of arrival* (direction d'arrivée)

GEVD : *Generalized Eigenvalue Decomposition* (décomposition selon les valeurs propres généralisée)

Dropout : Méthode de régularisation par abandon aléatoire de neurones

Early-stopping : Méthode de régularisation par interruption de l'apprentissage

HDF5 : *Hierarchical Data Format 5* (format de fichier avec arborescence)

ICASSP : *International Conference on Acoustics, Speech and Signal Processing*

LOCATA : *Acoustic Source Localization and Tracking* (challenge de localisation)

LSTM : *Long Short-Term Memory* RNN

RIR : *Room Inpulse Response* (réponse impulsionnelle de salle)

RNN : *Recurrent Neural Network* (réseau de neurones récurrent)

SIR : *Signal-to-Interference Ratio* (rapport signal à interférence)

SNR : *Signal-to-Noise Ratio* (rapport signal à bruit)

SMR : *Signal-to-Mixture Ratio* (rapport du signal d'intérêt sur le mélange)

SRIR : *Spatial Room Inpulse Response* (réponse impulsionnelle de salle spatiale)

TFCT : *Transformée de Fourier à Court Terme*

TPS : Équipe Orange / IMT / CCMB / CITY / TPS (Traitement de la Parole et du Son)

TR₆₀ : Temps de réverbération (à 60 dB)

VVM : *Velocity Vector Module* (cartographe utilisant le vecteur intensité)

WER : *Word-Error-Rate* (taux d'erreur de mots)

Chapitre 1

Présentation de l'entreprise

1.1 Le groupe Orange

Orange est une entreprise française issue de l'ancien groupe France Télécom. Elle est connue notamment pour son rôle de leader parmi les opérateurs européens et africains du mobile et de l'accès Internet : le groupe est implanté dans 28 pays avec 149000 salariés dans le monde, et propose des services de connectivité dans plus de 220 pays et territoires. Ainsi, avec plus de 263 millions de clients, Orange est la marque française de valeur la plus élevée au monde (51ème marque mondiale en 2018).

Bien que son cœur de métier historique soit les télécommunications, Orange est aujourd'hui une entreprise du numérique au sens large : le groupe propose à ses clients des services tels que l'accompagnement dans la transformation numérique des entreprises (menée par Orange Business Services) ou des acteurs de la santé (expertise e-santé d'Orange Healthcare), des contenus numériques (télévision, jeux, musique, cinéma, réalité virtuelle...) fournis par Orange Content, ou encore des services financiers tels que du commerce électronique (Orange Bank, Orange Money...).

De fait, le groupe mise donc énormément sur la recherche et l'innovation : son but est d' « inventer un futur dans lequel l'humain sera au centre de la révolution digitale », et consacre un budget de 700 millions d'euros à cette tâche.

1.2 Historique technologique

Les activités de recherche et développement (R&D) d'Orange ont commencé avec la fondation du Centre National d'Études des Télécommunication (CNET) en 1944, devenu France Télécom R&D en 2000 puis Orange Labs (2007). Voici quelques réalisations techniques marquantes qui en sont issues :

- 1961** : développement d'Antinéa, le plus puissant calculateur d'Europe utilisant la technologie des transistors, nouvelle à l'époque ;
- 1980** : test du Minitel, développé par le CNET, qui donnera lieu en 1982 au lancement commercial en France du premier ensemble au monde de services digitaux grand public ;
- 1981** : développement du réseau mobile numérique Marathon, qui alimentera, entre 1982 et 1991, la normalisation de la 2G (le GSM), impliquant en 1992 la commercialisation d'Itineris ;
- 2000** : contributions à la normalisation internationale de l'ADSL qui a permis l'émergence de l'Internet haut débit ;
- 2008** : conception de stations mobiles solaires déployées en Afrique et au Moyen-Orient ;
- 2009-2013** : contributions aux travaux de normalisation pour la compression de vidéo très haute qualité HEVC (*High Efficiency Video Coding*), qui a débouché en 2013 au CES à Las Vegas sur la 1ère démonstration mondiale d'un service « *Video on Demand* »utilisant HEVC ;
- 2015** : démonstration des progrès de la transmission optique suite à des travaux de recherche du projet SASER par un record du monde de liaison optique entre deux centraux opérationnels ;
- 2014-2016** : développement et participation à la normalisation pour la 5G, dont l'introduction est prévue pour après 2020.

1.3 Contexte du stage dans la hiérarchie d'Orange

La division Innovation, Marketing et Technologies¹ (IMT) est responsable de la mise en application des décisions prises par le Comité Exécutif en matière d'innovation. Les travaux de la division éclairent le regard du Comité sur le futur et nourrissent ainsi la réflexion stratégique du groupe. Ces activités sont réparties parmi plusieurs entités, dont notamment les Orange Labs : Orange Labs Recherche, Orange Labs Réseaux et Orange Labs Services portent la responsabilité technique globale des activités de recherche du Groupe, depuis la définition et la création des services innovants, jusqu'à la maintenance des solutions développées, et ce dans chaque pays.

Spécifiquement, *Orange Labs Services* (OLS) se focalise sur les technologies d'information, en visant la production efficace d'innovations de recherche qui se veulent simples et fiables pour les clients (d'où le terme de « services »), et génératrices de business pour le groupe. L'entité OLS opère dans le monde entier via les Orange Labs Internationaux (OLI), qui impliquent 25% de la production OLS en 2017. Sa gouvernance est divisée en directions métiers et thématiques, en plus des fonctions supports. En particulier, la direction thématique *Communication, Commerce and Mobile Banking* (CCMB) est en charge de la stratégie technique, de la conception, du développement, de l'intégration, du déploiement et de la maintenance des services de communication, du mobile banking et du commerce conversationnel.

Au sein de CCMB, le département *Communication, Immersion, Transaction & qualitY* (CITY) propose des solutions pour de nouvelles expériences de communication, d'immersion et de transaction (services audio et de communication immersifs, solutions voix et qualité de service, et autres services pour la sphère personnelle comme par exemple du banking).

1.3.1 L'équipe Traitement de la Parole et du Son

L'équipe *Technologies and Processing of Sound* (TPS), appellée « Traitement de la Parole et du Son » en français, fait partie du département CITY. Elle est en charge d'activités de recherche et développement pour les technologies audio pour divers contextes applicatifs : spatialisation sonore, interface homme-machine ... De fait, l'équipe travaille donc sur toutes les étapes de la chaîne audio : prise de son et séparation de sources, compression et codage, restitution, ou encore de la synthèse sonore. Elle possède une expertise contributive à l'établissement, au déploiement et au suivi des normes internationales dans ces domaines (MPEG, 3GPP, IETF).

TPS comprend une dizaine d'ingénieurs de recherche permanents, et autant d'employés sous contrats à durées déterminées (thésards, postdoctorants et stagiaires). L'équipe est répartie entre les deux centres Orange Labs de Lannion et Cesson-Sévigné. J'ai été accueilli par Alexandre Guérin et Lauréline Perotin, travaillant au centre de Cesson-Sévigné.

Ma place dans l'organigramme est donc la suivante : Orange / IMT / OLS / CCMB / CITY / TPS.

1. *Technology and Global Innovation* en anglais

Chapitre 2

La mission

2.1 Cadre et objectifs

2.1.1 Un Smart Speaker pour Orange

2.1.1.1 Qu'est-ce qu'un assistant personnel ?

Grâce aux récents progrès en traitement du langage naturel, la voix semble devenir la nouvelle interface privilégiée de l'interaction entre l'homme et la machine, en se proposant de remplacer les modes d'entrées textuelles classiques pour une multitude de services : navigation sur Internet, dictée de SMS, passage d'appels, pilotage de maisons connectées ...

De ce fait, des assistants personnels tels que le Google Home, l'Amazon Echo ou encore l'Apple HomePod commencent à entrer sur le marché.



FIGURE 2.1 – Exemples de *Smart Speakers*

L'utilisation de ces récents assistants suit le protocole suivant :

- Pour initier le dialogue avec l'assistant, l'utilisateur prononce un mot-clé, le *Wake-Up-Word*. Celui-ci est propre à l'assistant : « OK Google », « Alexa »... .
- Une fois l'utilisateur reconnu, un moteur de dialogue devient réceptif aux commandes de la part de ce dernier (phase d'écoute), et émet une réponse selon la commande reçue. Ce moteur de dialogue est notamment composé d'un module de transcription se chargeant de la reconnaissance automatique de la parole (*Automatic Speech Recognition*, ASR) et d'une unité de langage naturel (*Natural Language Unit*, NLU).

Ce protocole (en particulier la prononciation du mot-clé) doit être itéré à chaque commande.

En pratique, ces assistants sont constitués d'une enceinte connectée équipée de microphones pour la prise de son, d'où le nom de *Smart Speaker*, enceinte intelligente en français. Dans des conditions réelles d'utilisation, cette prise de son, et donc l'interaction entre l'utilisateur et l'assistant, sont perturbées par plusieurs obstacles, dont :

- Le bruit : en environnement domestique, l'assistant est susceptible d'être posé sur une table à côté d'une télévision par exemple, ou à proximité de toute source de bruit gênant la reconnaissance du mot-clé ou entravant le moteur de dialogue, comme d'autres locuteurs interférents ;
- La réverbération : les moteurs de reconnaissance vocale sont sensibles à la dégradation que l'acoustique des salles engendre sur les signaux de parole à capter ;

- Les échos : dus à la diffusion de contenus sonores par l'enceinte de l'assistant, ils peuvent masquer la voix de l'utilisateur et perturber le dialogue.

La Figure 2.2 est un exemple des modules de traitement embarqués dans un assistant pour mettre en place le protocole de reconnaissance, tout en limitant les perturbations précédemment décrites¹ :

- Annulateur d'écho : conserve le caractère audible de la voix de l'utilisateur, en atténuant l'écho ;
- *Beamforming* (formation de voie) : focalise la prise de son dans la direction de l'utilisateur de manière dynamique (ce qui réduit partiellement la réverbération et le niveau du bruit ambiant) ;
- Dé-réverbération : réduit les effets de l'acoustique de la salle, parfois avec un module indépendant du module de beamforming ;
- *Noise suppression* (suppression de bruit) : parfois implanté en tant que module indépendant, permet notamment de mieux segmenter les fins de phrases ;
- Détection du mot-clé : selon le temps mis par le système de formation de voie à se focaliser sur l'utilisateur, la détection peut se faire soit juste après l'annulation d'écho (flèches en pointillés), soit après les divers traitements évoqués dans cette liste (flèches en traits pleins).

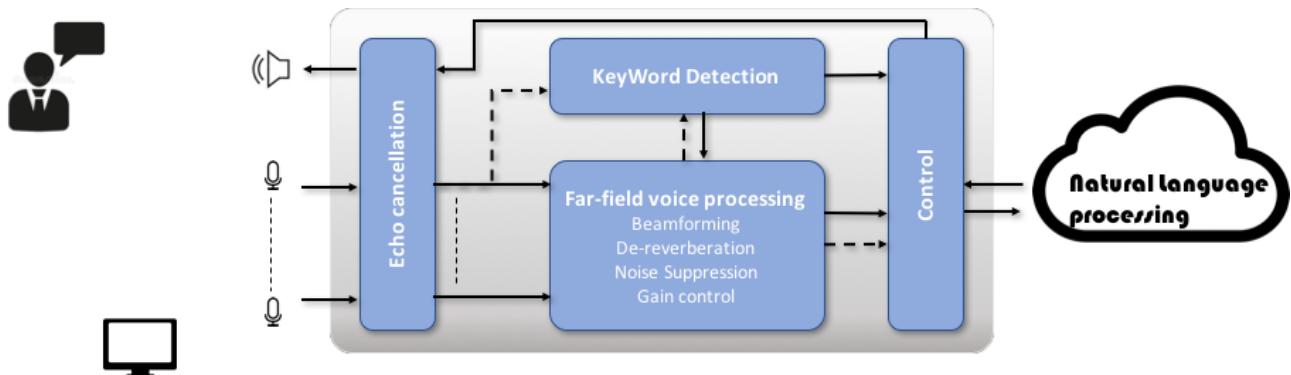


FIGURE 2.2 – Exemple de constitution d'un assistant domestique

2.1.1.2 Intérêt du stage pour Orange : Recherche et Développement pour Djingo

Dans ce contexte, Orange compte également proposer à ses clients de France son propre assistant virtuel multi-service, Djingo. Dans ce but, des activités de recherche et de développement sont menées par le groupe, avec les objectifs suivants :

- Développer un benchmark du produit en évaluant les solutions concurrentes sur le marché ;
- Évaluer les prototypes dans leurs versions successives ;
- Développer des algorithmes internes à Orange pour les implémenter dans les versions futures de Djingo, tout en indiquant les limites de fonctionnements des produits actuels.



FIGURE 2.3 – Djingo

Mon stage a pris place dans ce contexte, que je vais maintenant exposer, en développant notamment les enjeux et le cadre scientifique.

1. Chaque implémentation peut différer de ce modèle selon le fournisseur de solution.

2.1.2 Cadre scientifique : prise de son en champ lointain, localisation et séparation pour la reconnaissance automatique de la parole

Le principal challenge technique relevé par ces assistants vocaux est donc le traitement d'antenne mis en place lors de la prise de son effectuée grâce aux microphones de l'enceinte. L'antenne est définie par l'ensemble des microphones qui la composent, et notamment par sa géométrie : la figure 2.4 montre une antenne sphérique à 8 microphones utilisée par TPS.



FIGURE 2.4 – Antenne sphérique à 8 microphones

Le traitement d'antenne a lieu en plusieurs étapes essentielles :

- Pour un signal donné, chaque microphone va enregistrer une version modifiée (décalée dans le temps ou en amplitude) de ce signal source. Ces différents signaux nous donnent des informations sur les signaux sources et nous permettent notamment d'en estimer une localisation ;
- À partir de toutes les informations spatio-fréquentielles sur les sources, notamment leurs localisations, un filtrage spatial est effectué, afin d'obtenir une estimation du signal source en tant que combinaison linéaire des différents signaux enregistrés. C'est ce qu'on appelle le *beamforming*, formation de voie en français : le système se focalise sur les sources localisées.

Nous avons vu que ces traitements étaient problématiques, pour plusieurs raisons que nous allons détailler :

- La prise de son s'effectue en champ lointain, c'est à dire que l'utilisateur se trouve à une certaine distance de l'assistant. La voix de l'utilisateur n'est donc que rarement prépondérante devant les autres sources sonores, en présence d'interférences (locuteurs concurrents parlant aussi fort) ou de bruits environnants divers. Les signaux noyés dans le bruit sont durs à isoler ;
- De plus, la locution d'intérêt est sujette à la réverbération, c'est-à-dire l'effet du milieu de propagation des signaux, du notamment à la présence de surfaces (sol, murs) ou obstacles modifiant le signal émis. Les environnements réverbérants entravent la localisation des sources, indispensable au traitement d'antenne.

De plus, ce traitement d'antenne doit viser à optimiser la transcription pour la reconnaissance automatique de la parole. Les problématiques scientifiques mises en jeu pour un assistant vocal sont donc très larges, car impliquent de nombreux champs du traitement du signal audio et de la reconnaissance.

Ces domaines animent un vaste écosystème scientifique auquel prennent part les communautés académique et industrielle, à toutes les échelles dans le monde. Cela se traduit notamment par des challenges internationaux annuels comparant des techniques relevant de la « transcription automatique de la parole en prise de son lointaine », dont par exemple :

- CHIME (*Speech Separation and Recognition Challenge*, http://spandh.dcs.shef.ac.uk/chime_challenge/), qui s'intéresse à des algorithmes de séparation pour la transcription automatique dans ce cadre depuis maintenant 5 ans ;
- Le plus récent challenge LOCATA (*Acoustic Source Localization and Tracking*, <https://signalprocessingociety.org/get-involved/audio-and-acoustic-signal-processing/newsletter/locata-challenge/>), qui se préoccupe de l'estimation et du suivi de la localisation des sources, dans divers types de scènes sonores (une ou plusieurs sources, fixes ou mobiles) et avec divers systèmes de captation (antenne linéaire ou sphérique, fixe ou mobile).

Les résultats de ce type de challenge sont présentés lors de conférences réunissant les acteurs de ces communautés scientifiques. Un fait intéressant est l'irruption de méthodes impliquant des réseaux de neurones : ils sont utilisés soit en remplacement des approches standard de traitement du signal, soit en complément, notamment pour une estimation robuste au bruit de paramètres clés nécessaires au traitement d'antenne.

Ces enjeux sont aussi pris en compte par les industriels, qui veulent se positionner sur le marché et choisissent d'investir leurs départements R&D sur ces problématiques. Dans le cas d'Orange, des axes de recherche dédiés sont inclus dans la stratégie de l'entreprise, comme le projet ICCE (*Immersive Communication, Capture and Entertrainement*), dont l'objectif est le développement :

- de briques technologiques de cartographie et de séparation de sources en temps réel, utilisant le format ambisonique ; le choix d'intégrer des réseaux de neurones pour certaines tâches de traitement du signal impliquées, et de développer ces solutions utilisant TensorFlow, Keras et Python a été fait ;
- de prototypes en temps-réel, pour le transfert à des projets clients et benchmark.

Pour cela, de nombreuses ressources humaines sont mobilisées : en plus des internes impliqués sur le sujet, le groupe recrute aussi des doctorants, qui développent des solutions dans le cadre de leurs thèses CIFRE. On peut par exemple citer les 3 derniers doctorants supervisés par Alexandre Guérin :

- Zaher El Chami a soutenu en 2010 une thèse sur le rehaussement de la parole par des techniques de séparation de sources bi-capteurs, et a également présenté une nouvelle méthode de comptage et de localisation de sources dans ce cadre ;
- Mathieu Baqué, ayant soutenu en 2017, a travaillé sur l'analyse de scène sonore multi-capteurs [1], et a mis au point un outil de manipulation temps-réel en environnement réverbérant de contenus audio captés au format ambisonique ;
- Lauréline Péroton, dont la seconde année de thèse est en cours, travaille sur l'intérêt des réseaux de neurones profonds pour ces tâches de localisation et de séparation de sources pour les contenus ambisoniques.

Durant ces thèses, des stagiaires viennent parfois compléter l'équipe pour mener des activités de recherche et développement complémentaires, comme Andrea Vaglio, qui a travaillé avec Mathieu en 2016, ou bien moi-même dans le cadre de la thèse de Lauréline.

2.2 Objectifs du stage

Ce stage est une contribution au développement de deux modules, un de localisation, et un de séparation.

2.2.1 Localisation : Réécriture du *Velocity Vector Module* (VVM) en Python

Le *Velocity Vector Module* (VVM) est un logiciel de cartographie développé par Alexandre, Srđan et Arnaud de l'équipe TPS. Il s'agit d'un code de recherche initialement développé en MATLAB. Or, pour les raisons suivantes, une version de l'algorithme en Python était nécessaire :

- Pour les prototypes de systèmes embarqués comme les assistants vocaux, les codes de recherches sont traduits en langage ANSI C. Pour une éventuelle conversion (et pour la maintenance) de ces algorithmes, un langage orienté-objet se montre plus simple à manipuler ;
- Le VVM est supposé tourner en temps-réel afin d'effectuer la localisation en direct. Pour l'implémenter de façon optimale, utiliser des boards de développement est nécessaire, et ceux-ci intègrent des interpréteurs Python, pour faire tourner en temps-réel du code ;
- Comme précédemment évoqué, le traitement d'antennes fait de plus en plus appel à des réseaux de neurones profonds, et TPS suit cette tendance. Les algorithmes d'apprentissage développés par l'équipe pour ces tâches sont implémentées avec Python, TensorFlow et Keras. Ainsi, pour des raisons d'unification de tous les algorithmes de recherches de l'équipe, implémenter du code en Python orienté objet est devenu nécessaire. Cela implique de la modularité : on peut intégrer plus facilement une partie d'un code de recherche à un autre et interfaçer plus simplement les différentes briques de développement.

Cette mission, encadrée par Alexandre et Srđan, m'a occupé les 7 premières semaines de stage.

2.2.2 Séparation : Reprise et poursuite des travaux sur le système de Lauréline

Dans le cadre du développement de Djingo précédemment introduit, Lauréline a soumis avec ses encadrants de thèse un article [17] à la conférence ICASSP (*International Conference on Acoustics, Speech and Signal Processing*), paru en Avril 2018, dans lequel elle présente le système de séparation de sources qu'ils ont mis au point. Ce système étant encore en cours de développement lors de la soumission, plusieurs pistes d'amélioration étaient encore à envisager afin d'optimiser l'efficacité et les performances de la méthode.

Le but principal de mon stage a donc été d'explorer certaines des pistes de recherches que Lauréline a mis de coté pendant ce développement. Il s'agissait principalement d'étudier l'influence de certains hyper-paramètres et de tester différentes architectures de réseau de neurones. Pour cela, une phase de développement a été requise pour passer l'ancien code en orienté objet, et en profiter pour intégrer diverses améliorations pour rendre le code plus modulaire et plus performant. La totalité de cette partie, en collaboration avec Lauréline et Alexandre, a duré environ 4 mois, de la fin de la transcription du VVM à la fin du stage (hors rédaction du rapport).

Chapitre 3

Le *Velocity Vector Module* (VVM)

3.1 Format ambisonique et environnement acoustique

Comme évoqué dans le cadre, la localisation est la première étape nécessaire à la séparation de sources. Le principe de la localisation est d'exploiter les propriétés acoustiques de l'onde sonore au cours de sa propagation. Le formalisme choisi pour représenter le champ acoustique est ici le format ambisonique. Il est basé sur la décomposition en harmoniques sphériques, et sera adopté pour tout le stage. Nous allons l'introduire dans cette première partie, avec toutes les notions nécessaires relatives à l'environnement acoustique.

3.1.1 Représentation du champ acoustique

3.1.1.1 Équation des ondes

Considérons l'air libre (sans source) comme un fluide parfait au repos, homogène. Nous ne considérons que les variations de pression de l'air dues aux phénomènes acoustiques. Sous l'hypothèse que ces variations soient faibles, le champ de pression acoustique p vérifie en tout point M de l'espace l'équation des ondes, dite équation de d'Alembert : $\Delta p_M - \frac{1}{c_0^2} \frac{\partial^2 p_M}{\partial t^2} = 0$, où c_0 est la célérité du son dans l'air, t le temps et Δ l'opérateur Laplacien.

Abordons la résolution de cette équation pour un signal harmonique. En effet, par décomposition de Fourier, tout signal temporel est décomposable en somme de signaux sinusoïdaux. La résolution pour les signaux harmoniques mène donc au cas général par théorème de superposition.

3.1.1.2 Résolution en coordonnées sphériques

Pour résoudre l'équation des ondes, on associe à l'espace un repère de coordonnées sphériques centré sur l'auditeur¹. Notons respectivement r , θ et ϕ la distance entre le point M et le centre du repère, l'azimuth et l'élévation du point M. Dans tout le rapport, ces angles seront donnés en degrés.

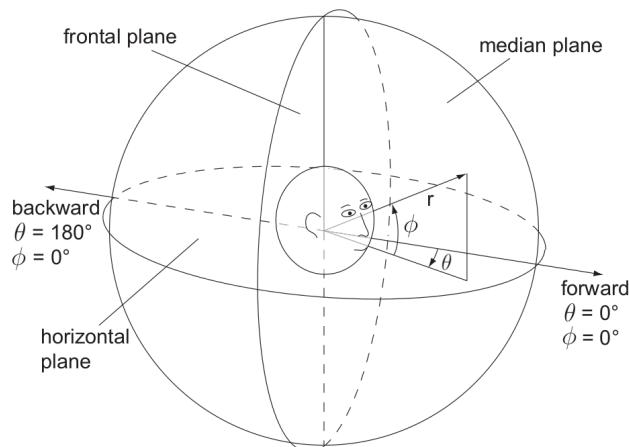


FIGURE 3.1 – Système de coordonnées sphériques céphalo-centré. Source : [13]

1. Le centre du repère est situé entre les deux oreilles.

Avec ces notations, pour le signal harmonique considéré $p(r, \theta, \phi, \omega, t) = P(r, \theta, \phi)e^{i\omega t}$, l'équation des ondes devient l'équation de Helmholtz, qui admet une solution à variables séparées de la forme :

$p(r, \theta, \phi, \omega, t) = R(r)\Theta(\theta)\Phi(\phi)e^{i\omega t}$, où $R(r)$, $\Theta(\theta)$ et $\Phi(\phi)$ admettent des formes fermées connues. En supposant que le signal provienne de l'infini, et passant les développements calculatoires (disponibles dans [1] par exemple), on obtient finalement la représentation générale du champ de pression acoustique suivante :

$$p(r, \theta, \phi, \omega, t) = \sum_{m=0}^{\infty} \sum_{n=0}^m \sum_{\sigma=\pm 1} A_{mn}^{\sigma} \tilde{Y}_{mn}^{\sigma}(\theta, \phi) i^m j_m(kr) e^{i\omega t}$$

où :

- $\tilde{Y}_{mn}^{\sigma}(\theta, \phi)$ représente les fonctions harmoniques sphériques normalisées de degré $m \geq 0$ et d'ordre $n \leq 0$, avec $\sigma = \pm 1$. Il s'agit de polynômes trigonométriques en θ et ϕ , dont l'expression implique les polynômes de Legendre. La représentation graphique des premières fonctions harmoniques sphériques est donnée en figure 3.2, avec en rouge les valeurs positives, et en bleu les valeurs négatives. Chaque fonction harmonique sphérique représente une directivité donnée de l'espace ;
- A_{mn}^{σ} est une constante s'interprétant comme une captation microphonique dans une directivité donnée, correspondant donc à une harmonique sphérique ;
- $j_m(kr)$ est une fonction de Bessel sphérique, k étant un entier ;
- i est le complexe imaginaire pur de carré égal à -1.

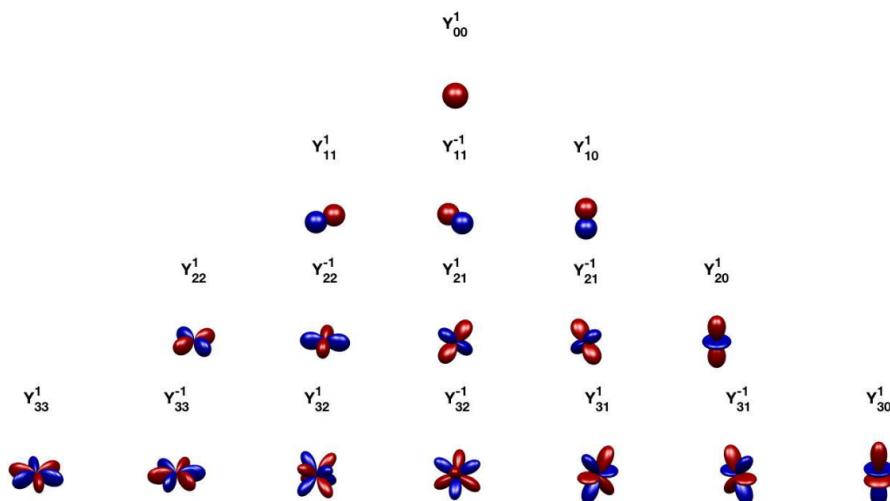


FIGURE 3.2 – Fonctions harmoniques sphériques de degré 0 (première ligne) à 3 (dernière ligne). Source : [1]

Avec cette dernière équation et le principe de superposition, on peut ainsi obtenir, dans le cadre précédemment introduit, la décomposition exacte théorique de tout champ acoustique grâce à une infinité d'observations coïncidentes ayant pour directivités les fonctions harmoniques sphériques. Il existe d'autres formalismes avec des notations différentes. Celui introduit par Mathieu dans sa thèse [1] a été largement repris ici, entre autres car j'ai commencé à appréhender l'ambisonie avec la lecture du début de cette référence.

3.1.2 Format B : ambisonie à l'ordre 1

De fait, l'expression trouvée dans la partie précédente est inutilisable à cause de l'infinité de termes à calculer, d'où la nécessité de l'approximer en n'utilisant qu'une partie des harmoniques sphériques. On parle de troncature à l'ordre M (il est d'usage de parler d'harmoniques d'ordre M plutôt que de degré M lorsque l'ordre et le degré ne sont pas évoqué simultanément [1, 5]).

L'expression du champ acoustique tronquée à l'ordre M devient :

$$p(r, \theta, \phi, \omega, t) = \sum_{m=0}^M \sum_{n=0}^m \sum_{\sigma=\pm 1} A_{mn}^{\sigma} \tilde{Y}_{mn}^{\sigma}(\theta, \phi) i^m j_m(kr) e^{i\omega t}$$

Plus M est grand, plus on considère d'harmoniques, et plus l'approximation du champ sonore est précise.

Le format considérant les premiers ordres ambisoniques 0 et 1 est appelé format B. Quatre canaux constituent ce format : la composante d'ordre 0, notée W , est la composante omnidirectionnelle du champ sonore, tandis que les composantes X , Y et Z , formant l'ordre 1, sont assimilables à des gradients de pression orientés suivant les trois dimensions de l'espace. En notant $Y_{mn}^\sigma = \tilde{Y}_{mn}^\sigma / \sqrt{2m+1}$ les fonctions harmoniques sphériques de degré m et d'ordre n non normalisées, les équations suivantes caractérisent le format B :

$$\begin{bmatrix} W \\ X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} Y_{00}^1(\theta, \phi)p \\ Y_{11}^1(\theta, \phi)p \\ Y_{11}^{-1}(\theta, \phi)p \\ Y_{10}^1(\theta, \phi)p \end{bmatrix} = \begin{bmatrix} p \\ \sqrt{3} \cos \theta \cos \phi \cdot p \\ \sqrt{3} \sin \theta \sin \phi \cdot p \\ \sqrt{3} \sin \phi \cdot p \end{bmatrix} = \begin{bmatrix} 1 \\ \sqrt{3} \cos \theta \cos \phi \\ \sqrt{3} \sin \theta \sin \phi \\ \sqrt{3} \sin \phi \end{bmatrix} p = \mathbf{d}(\theta, \phi) \cdot p \quad (3.1)$$

en notant $\mathbf{d}(\theta, \phi)$ le vecteur directionnel pointant vers une source localisée par ses coordonnées (θ, ϕ) , qui représente donc la *direction d'arrivée* de la source.

3.1.3 Environnement acoustique

Les expressions précédentes sont valables en champ libre, ce qui n'est pas le cas dans les cas réels. En effet, la présence d'un sol ou d'autres surfaces réfléchissantes (murs, plafond ...), ou d'autres d'obstacles perturbe la propagation des ondes sonores. En environnement fermé, l'interaction entre des ondes sonores et les éléments précédents est appelée effet de salle. Cet effet de salle doit être modélisé pour que notre cadre reste valide.

3.1.3.1 Modélisation de l'effet de salle (SRIR)

On modélise l'effet de salle avec la donnée d'une réponse impulsionnelle entre une source sonore et un point de mesure, que l'on supposera ici confondu avec l'origine du repère. On désigne les réponses impulsionnelles de salle par leur acronyme anglais : RIR signifie *Room Impulse Response*. Dans le cas le plus simple, une RIR est omnidirectionnelle. Si le capteur servant à la mesure est directionnel, alors celui-ci peut encoder plusieurs directivités, et la réponse impulsionnelle sera dite spatiale : on parle alors de SRIR, *Spatial Room Impulse Response*. Dans notre cas, nous manipulons des signaux enregistrés avec un microphone ambisonique encodant le format B précédemment introduit. Les SRIR que nous manipulerons durant tout le stage et le rapport sont donc des réponses multicanales, dont chaque canal est une RIR. Nous noterons respectivement W , X , Y et Z les quatre canaux de ces SRIR, en cohérence avec les notations du format B.

Une RIR se décompose en trois parties :

- Le champ direct (*direct path*), empruntant le trajet source - microphone sans réflexion ;
- Les premières réflexions (*early echoes*), du son sur les parois ;
- La réverbération, modélisée comme un champ diffus de densité d'énergie acoustique uniforme.

Plusieurs indices permettent de caractériser une RIR, parmi lesquels le temps de réverbération TR_{60} , correspondant au temps mis par l'énergie acoustique pour décroître de 60 dB dans la salle en l'absence d'excitation. Ces notions sont illustrées sur la figure 3.3.

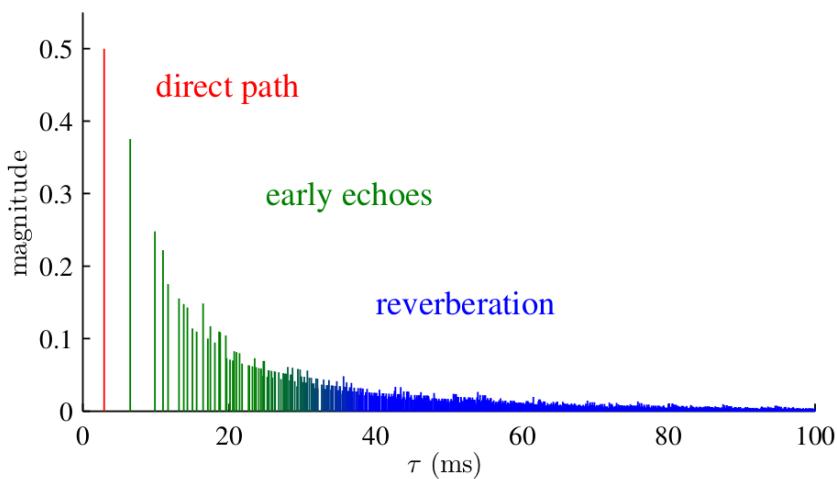


FIGURE 3.3 – Illustration schématique d'une SRIR de TR_{60} valant 0.25 seconde. Source : [8]

3.1.3.2 Intensité acoustique

L'effet de salle précédemment décrit modifie la direction de propagation de l'énergie acoustique, qui est synthétisée par la notion d'intensité acoustique. Notons \vec{v} la vitesse particulière ou vélocité² du point M considéré, alors l'intensité acoustique instantanée $\vec{I}(t) = p(t)\vec{v}(t)$ représente le flux d'énergie acoustique par unité de surface. Sa moyenne temporelle, l'intensité active $\vec{I}_a = \Re\{p \cdot \vec{v}^*\}$, décrit donc le flux d'énergie acoustique se propageant à travers une surface élémentaire.

On se référera à 3.2.1 ou par exemple [1] pour un développement plus long sur ces notions.

3.2 Description de l'algorithme

Comme évoqué en 3.1, le principe de la localisation est d'utiliser des propriétés liées à la propagation des ondes sonores. L'énergie acoustique en est un exemple, et nous venons de voir que l'intensité acoustique résume l'information de variation de cette énergie. L'algorithme du *Velocity Vector Module*, que l'on appelle VVM, utilise donc l'intensité acoustique pour en déduire les *directions d'arrivées* des sources sonores dans un contenu audio au format B.

Il s'agit d'un logiciel fonctionnant en temps réel, chacune des étapes que l'on va décrire dans les parties 3.2.1 et 3.2.2 est ainsi incluse dans une boucle temporelle : à chaque trame temporelle t , ces étapes sont effectuées dans cet ordre, et la boucle est réitérée jusqu'à la fin de l'enregistrement.

Notons \mathbf{S} l'enregistrement ambisonique considéré. Ce signal multicanal sera traité par une analyse temps-fréquence que l'on va effectuer avec la Transformée de Fourier à Court Terme (TFCT, voir 2.3 de [14] pour détails). Dans toute la suite du rapport, on appellera « domaine transformé » le domaine temps-fréquence associé au domaine temporel par TFTC. Nous pouvons donc noter pour toute la suite :

$$\mathbf{S}(t, f) = \begin{bmatrix} \mathbf{S}_W(t, f) \\ \mathbf{S}_X(t, f) \\ \mathbf{S}_Y(t, f) \\ \mathbf{S}_Z(t, f) \end{bmatrix} = \begin{bmatrix} W(t, f) \\ X(t, f) \\ Y(t, f) \\ Z(t, f) \end{bmatrix}$$

t désignant la trame temporelle et f la fréquence.

3.2.1 Vecteur intensité

Le fondement de l'algorithme est le calcul, à chaque trame t , du vecteur intensité instantanée $\vec{I}(t) = p(t)\vec{v}(t)$. Dans le domaine transformé, en ne considérant que la partie réelle (appelée partie active) de l'intensité, cette expression devient :

$$\vec{I}(t, f) = \begin{bmatrix} I_X(t, f) \\ I_Y(t, f) \\ I_Z(t, f) \end{bmatrix} = \begin{bmatrix} \Re\{W(t, f)^* X(t, f)\} \\ \Re\{W(t, f)^* Y(t, f)\} \\ \Re\{W(t, f)^* Z(t, f)\} \end{bmatrix}$$

Cela représente autant de vecteurs que de points temps-fréquence pour \vec{I} , et donc autant de directions d'arrivées par (3.1). Dans les faits, au lieu d'utiliser ces équations et des relations trigonométriques trop couteuses en calcul pour remonter aux coordonnées (θ, ϕ) , les différentes directions d'arrivées possibles sont discrétisées par un ensemble fini de couples (θ, ϕ) ordonnés, formant une sphère d'une taille définie selon les règles de Lebedev.

D'un point de vue analytique, cela pourrait sembler suffisant pour fournir une localisation instantanée des sources j . Cependant, comme exposé dans [6], ou en 2.4.6 dans [13], en raison des mouvements des sources et de la réverbération induite par l'effet de salle, ces estimations sont beaucoup trop fluctuantes dans le domaine temps-fréquence pour être utilisables en l'état.

Pour limiter le problème de ces fluctuations spectrales, on réalise l'agrégation de points temps-fréquences pour effectuer la localisation sur des histogrammes spatiaux lissés, plus stables que sur chaque trame.

2. *Velocity Vector* en anglais, d'où le nom du VVM : *Velocity Vector Module*

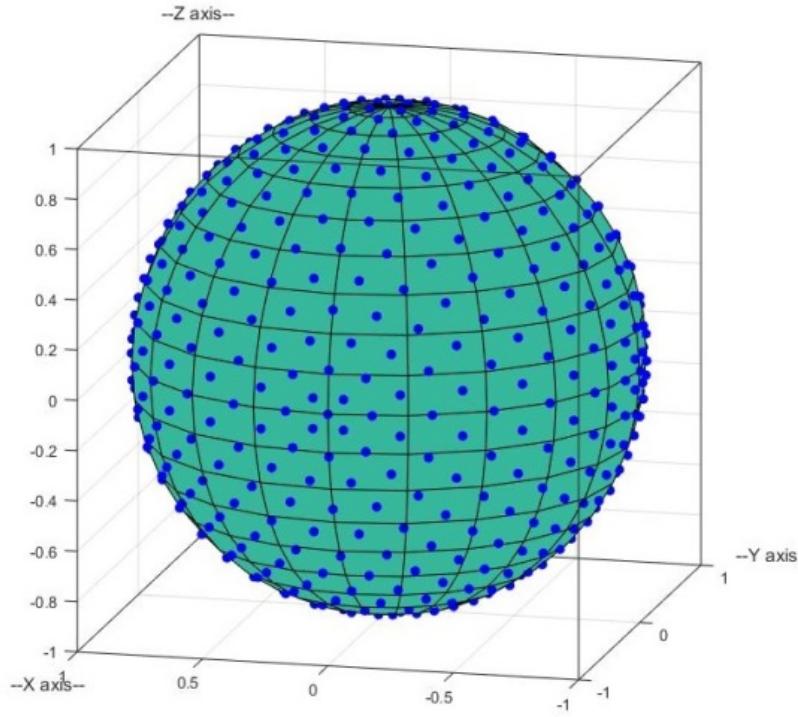


FIGURE 3.4 – Discréétisation des directions d’arrivées détectables par le VVM

3.2.2 Histogrammes

Pour la construction de ces histogrammes agrégés, un tampon (*buffer*) est utilisé pour conserver en mémoire les derniers histogrammes instantanés pour le lissage. Il est mis à jour à chaque itération. À chaque trame, l’analyse temps-fréquence fournit plusieurs valeurs de $\tilde{I}(t, f)$ pour chaque fréquence indexée. La discréétisation précédente permet d’obtenir une localisation instantanée $(\theta(t), \phi(t))$ pour indexer le contenu spectral de $\tilde{I}(t, f)$. À partir des histogrammes instantanés contenus dans le tampon, un histogramme lissé est calculé. On peut constater sur la figure 3.5, sur lesquels les points chauds sont supposés correspondre aux localisation des sources vers lesquels pointent les vecteurs directeurs, que les valeurs instantanées sont inutilisables, alors que les histogrammes lissés permettent de distinguer deux sources ici.

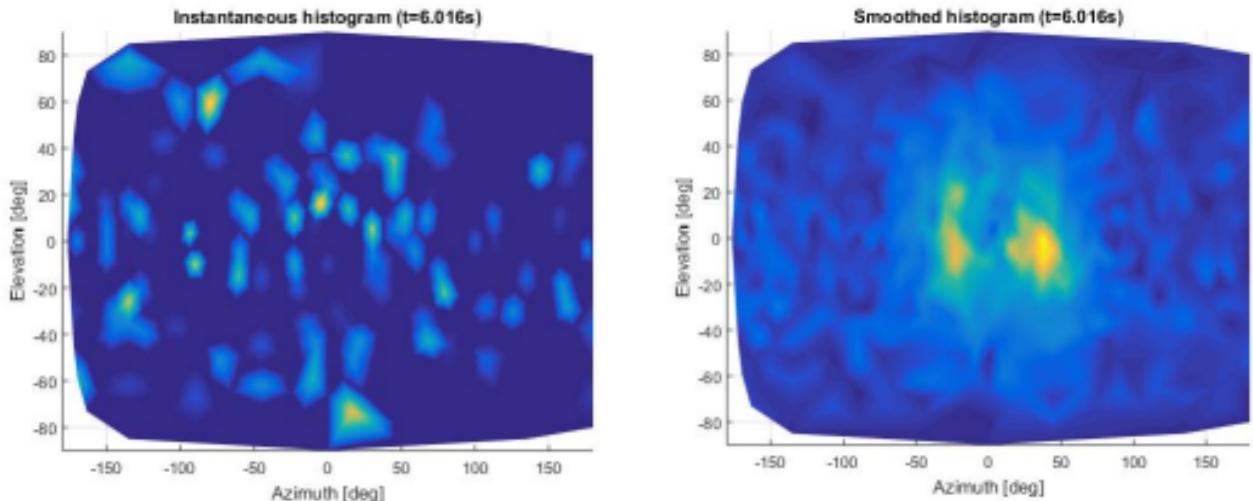


FIGURE 3.5 – Comparaison des histogrammes spatiaux instantané et lissé

La déduction des coordonnées $(\theta(t), \phi(t))$ à partir de ces histogrammes lissés se fait par projection sur les dimensions angulaires, grâce à un système de suivi conditionné par la détection d'activité vocale dans la trame.

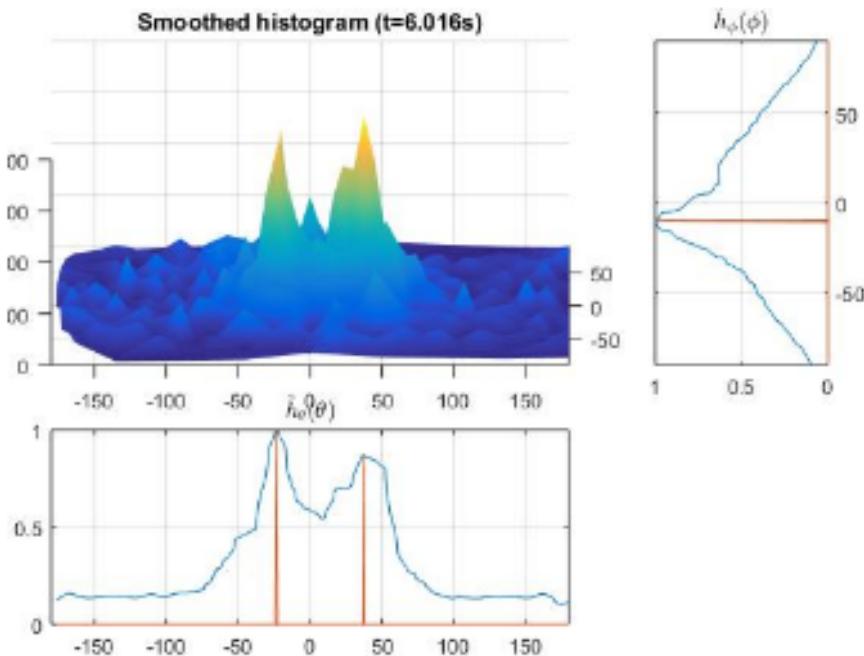


FIGURE 3.6 – Un histogramme en 2D et ses projections en 1D

3.3 Implémentation

3.3.1 Comparaison MATLAB / Python

3.3.1.1 MATLAB

MATLAB est à la fois un langage de programmation et un environnement de développement intégré, conçu spécialement pour le calcul numérique et commercialisé par la société MathWorks. Il s'agit d'un langage faiblement typé, dont l'utilisation repose essentiellement sur la manipulation de matrices³. Son but est de rendre intuitives et efficaces la manipulation et la visualisation de données, en permettant de faire de l'expérimentation efficiente pour tout type de science. Grâce à ses performances, notamment en temps réel (<https://fr.mathworks.com/products/matlab/matlab-execution-engine.html>), et grâce à ses *toolboxes* (boîtes à outils), il est souvent le langage idéal pour le développement de solutions à but non commercial⁴.

Ainsi, indépendamment des motifs invoqués en 2.2.1, MATLAB constituait une outil assez naturel pour la première approche du développement d'un code de recherche comme le VVM. En effet, bien qu'ayant été breveté, il s'agit d'un algorithme interne, qui ne sera pas utilisé en l'état dans Djingo⁵. De plus, le développement du VVM s'est poursuivi sur la version Python du code que j'ai livré.

3.3.1.2 Python

Python est un langage de programmation objet, généraliste et multiparadigme. C'est un langage de haut niveau, à fort typage dynamique. Il inclut une large bibliothèque standard couvrant notamment les problématiques du traitement des chaînes de caractères, des protocoles réseaux, de l'ingénierie logicielle et de l'interface avec les systèmes d'exploitation (<https://docs.python.org/fr/3/faq/general.html>).

De plus, étant placé sous licence libre, il est extensible grâce aux bibliothèques créées par une large communauté de développeurs. C'est notamment grâce aux bibliothèques NumPy (*Numerical Python*), SciPy (*Scientific Python*) et Matplotlib que Python a été étendu au calcul numérique et à la visualisation de données.

3. Le nom MATLAB vient de l'anglais *Matrix Laboratory*.

4. Les *toolboxes* étant mises au point et commercialisées par MathWorks, revendre un produit développé avec une de ces *toolboxes* implique des problèmes de licences complexes, onéreux et rarement rentables.

5. Les codes de recherches implémentés par TPS sont de toutes façons traduits en langage C, plus adapté aux systèmes embarqués.

Portable et libre, il a aussi des avantages pour les usages de TPS. En particulier, TensorFlow est une bibliothèque de *machine learning* compatible avec Python, et Keras [3] en est une interface de haut niveau, plus proche de l'état d'esprit du langage Python, qui se veut plus simple et intuitif que la plupart des autres langages de programmation [18]. Ces raisons, avec celles évoquées en 2.2.1, ont justifié l'intérêt de cette traduction.

3.3.1.3 Quelques différences fondamentales

Indexation : Une différence clé entre les deux langages est que MATLAB commence ces numérotations à partir de 1, alors que Python commence à partir de 0. La première option est plus naturelle pour les humains, relativement à leur façon de manipuler les concepts mathématiques (et notre manière de raisonner tout simplement) : la première dimension d'une matrice sera numérotée 1, la seconde 2...

Cependant, Python utilise une indexation qui se veut plus utile pour simplifier les traitements algorithmiques appliqués aux séquences manipulées. Les références suivantes tentent d'illustrer ce dernier fait :

<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html>

<https://groups.google.com/forum/?hl=en#!topic/comp.lang.python/xfH2pQCH8iY%5B76-100%5D>

Gestion des types : De par la nature des langages utilisés, les versions MATLAB et Python de l'algorithme n'utilisent pas le même type d'objets pour manipuler les signaux et leurs traitements, et ils ne sont pas implémentés de la même manière. La différence fondamentale ici est que les variables de MATLAB ne sont globalement que des nombres, vecteurs et matrices, concepts mathématiques à manipuler comme on le ferait en algèbre linéaire, alors que *toutes* les variables de Python sont des objets comprenant méthodes et attributs différents. Cette différence se traduit par exemple dans la gestion des booléens par les deux langages : tandis que MATLAB utilise simplement 1 et 0 pour les valeurs respectives de *Vrai* ou *Faux*, Python implémente des types d'objets spécifiques pour ces booléens.

Diverses ressources comparatives pour les deux langages sur ce plan existent, dont <https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html>.

Bibliothèques et fonctions existantes : Ici, le principe est le même pour les deux langages : ils sont extensibles grâce à des *toolboxes* ou bibliothèques annexes. Cependant, le caractère commercial des *toolboxes* implique un développement et un support professionnel qui favorise davantage la qualité de ces extensions. Par exemple, les *toolboxes* MATLAB de traitement du signal audio sont irréprochables, alors que les bibliothèques Python sont beaucoup plus *ad hoc* : on notera la difficulté à trouver une bibliothèque Python lisant les fichiers WAV encodés en 24 bits.⁶ On peut faire le même constat en ce qui concerne les tracés et interfaces graphiques : la bibliothèque Matplotlib de Python s'inspire d'ailleurs largement de MATLAB, mais en concédant un certain retard.

3.3.2 Architecture du code objet

Mon implémentation du VVM repose essentiellement sur une fonction principale *loc_multisource()* et sur deux classes, *RawSignal* et *QuantizedSignal*. De plus, une fonction permet de tracer en temps réel les histogrammes, comme illustré en figures 3.5 et 3.6

3.3.2.1 Fonction principale : *loc_multisource()*

La fonction *loc_multisource()* reçoit en arguments les paramètres du problème, dont notamment l'enregistrement ambisonique **S** à traiter, les paramètres de la TFCT, les filtres et le tampon utilisés pour lisser les histogrammes, ou encore le nombre de points de discréttisation à utiliser pour la sphère de Lebedev. Elle retourne, sous la forme de listes θ_hat et ϕ_hat , les localisations estimées à chaque trame temporelle t de la boucle.

Au début de l'exécution, les paramètres donnés en arguments sont comparés à des valeurs par défauts définies en tant qu'attributs des classes *RawSignal* et *QuantizedSignal*. La fonction *loc_multisource()* utilise la valeur par défaut d'un paramètre si aucune valeur n'est spécifiée en argument.

Après la paramétrisation, un objet *raw_sound* de la classe *RawSignal* est instancié à partir de ces paramètres et de l'enregistrement **S**.

On entre ensuite dans la boucle temporelle principale, dans laquelle l'algorithme décrit en 3.2 est exécuté.

6. De même, j'ai dû implémenter moi-même en Python l'équivalent de la convolution circulaire de MATLAB
<https://fr.mathworks.com/matlabcentral/fileexchange/1117-circonv>, indisponible dans la bibliothèque de référence SciPy.

3.3.2.2 Classe *RawSignal* : signal avant discréétisation

La méthode principale de la classe est son `_init_()` : il s'agit d'une méthode spéciale de Python, appelée systématiquement lors de l'instanciation d'un objet. Pour cette classe, il s'agit de préparer le calcul instantané de $\vec{I}(t, f)$. Les quatre canaux ambisoniques de \mathbf{S} sont stockés dans des matrices Python, puis segmentés pour la TFCT. Les signaux segmentés, ainsi que la bande de fréquence et autres paramètres de la TFCT, sont stockés en tant qu'attributs de l'instance `raw_sound`. De même, les paramètres nécessaires à la détection d'activité vocale dans chaque trame du signal sont stockés en tant qu'attributs.

Une fois dans la boucle principale, à un instant t donné, les coordonnées du vecteur intensité $\vec{I}(t, f)$ sont calculées comme en 3.2.1, et également stockées en tant qu'attributs de `raw_sound`. Ces attributs sont mis à jour à chaque trame temporelle t .

3.3.2.3 Classe *QuantizedSignal* : signal après discréétisation

Une fois le vecteur intensité $\vec{I}(t, f)$ calculé, nous le discréétisons selon les règles de Lebedev. C'est la méthode `_init_()` de la classe *QuantizedSignal* qui s'en charge : elle calcule l'indice du point de la sphère ordonnée représentant la direction d'arrivée correspondant à $\vec{I}(t, f)$. Cet indice $ind_{\theta, \phi}$ est stocké en tant qu'attribut de l'instance `quantized_signal` de la classe.

Cet objet dispose de plusieurs méthodes, exécutant séquentiellement les différentes étapes de l'algorithme :

- Une méthode `hist_2D()` gère les histogrammes en deux dimensions. Grâce à `quantized_signal.ind_{\theta, \phi}(t)` et au tampon gardant en mémoire les histogrammes instantanés précédents, un histogramme lissé est calculé à chaque trame t , et stocké en tant qu'attribut de `quantized_signal` ;
- Une méthode `hist_1D()` se charge de construire un histogramme pour chaque dimension angulaire, par projection de l'histogramme à deux dimension précédent (voir figure 3.6). De même, ces histogrammes sont stockés en tant qu'attributs de `quantized_signal` ;
- Une méthode `localisation()` effectue ensuite l'estimation effective des localisations $\theta_hat(t)$ et $\phi_hat(t)$, à partir d'un ensemble de règles conditionnées par la détection d'activité vocale et le suivi des valeurs précédentes des histogrammes ;
- La méthode `localisation()` fait pour cela appel à une autre méthode, `find_2D_coord_along_1D()`, qui permet, à partir des valeurs de deux histogrammes unidimensionnels, de remonter au couple de coordonnées $(\theta_hat(t), \phi_hat(t))$. Ces valeurs sont stockées dans les listes `\theta_hat` et `\phi_hat` définies lors de l'initialisation de `loc_multisource()`.

3.4 Conclusion

Cette première partie de stage a été pour moi une introduction très satisfaisante et instructive. D'une part, ce fut l'occasion de découvrir d'un point de vue théorique quelques fondamentaux du format ambisonique et du traitement du signal audio, notamment grâce à la lecture d'articles comme [6], et des premières parties des thèses [1] et [13]. D'autre part, j'ai vu une application concrète d'une application de certains de ces principes au sein d'un code de recherche complexe. Cette transcription m'a fait progresser sur plusieurs points :

- La compréhension de certains détails de l'algorithme a soulevé des questions, dont la résolution a augmenté mes connaissances en traitement du signal, en plus de me fournir mes premières bases d'acoustique ;
- J'ai gagné du recul, de l'expérience et des connaissances à la fois sur l'utilisation de MATLAB et de Python. Accomplir cette mission a nécessité que je progresse quotidiennement sur les deux langages, et même au-delà. En effet, j'ai du apprendre à distinguer clairement les notions théoriques mises en jeu et leurs implémentations informatiques. Ce changement de point de vue a changé ma manière d'appréhender une mission de développement dans un cadre scientifique, et je pense que je serai désormais plus efficace que je ne l'ai été auparavant sur des missions analogues ;
- Dans le même sens, c'est ma première expérience de projet en orienté objet d'une aussi grande ampleur, sur lequel je suis parti d'une « feuille blanche ». J'ai eu l'occasion d'appliquer plus concrètement des concepts clés vus pendant ma scolarité à Télécom SudParis, en plus de gagner une expérience supplémentaire en développement. J'ai eu l'occasion de faire la synthèse de tous mes acquis et d'aller au-delà en acquérant les connaissances nécessaires à l'exécution de ma mission ;
- La partie « séparation multicanale » du stage a également nécessité une phase de réécriture de code, durant laquelle j'ai dû changer le paradigme globalement procédural du code de Lauréline, pour lui conférer un caractère orienté objet plus clair et plus modulaire. Ce travail sur le VVM m'a en partie préparé à cette autre tâche, en remettant en place dans mon esprit de nombreux concepts.

De plus, j'en tire la satisfaction personnelle d'avoir effectué un travail utile, dont j'ai vu les fruits avant même mon départ. En effet, malgré sa faible latence et ses faibles besoins en ressources de calcul, cette version du VVM admet certaines limites, dont un manque de robustesse à la réverbération et un nombre de cibles simultanées limité à deux. Peu de temps après que je livre cette version Python du VVM, Alexandre et Srđan ont commencé à travailler sur une mise à jour de l'algorithme, levant ces problèmes grâce à un système de filtrage particulaire remplaçant les histogrammes agrégés. Alexandre et Srđan ont commencé l'implémentation de cette nouvelle version du VVM sur la base du code que j'ai rendu.

Enfin, d'un point de vue plus introspectif, cela m'a donné un aperçu assez inédit de ce que j'étais capable d'apprécier sur un plus long terme que la plupart de mes projets scolaires. C'est en effet la transcription du VVM qui m'a indiqué à quel point j'étais susceptible d'apprécier le développement informatique : je suis prêt à y consacrer des heures, à la forte condition que l'objectif final m'intéresse. C'est ma curiosité relative à chacune des briques du traitement qui m'a autant motivé à avancer sur cette partie de la mission : je me suis surpris, lorsque j'avais saisi une partie de l'algorithme, à refuser de m'arrêter de coder avant de l'avoir implémentée. Cette mission m'a aussi donné un aperçu de certaines de mes capacités : bien qu'ayant quelques bases de traitement du signal et de programmation nécessaires à l'exécution de cette mission, j'ai dû apprendre sur le moment énormément de petites choses qui étaient trop floues dans mon esprit, ou qu'il me manquait tout simplement.

Je ne peux pas tout savoir avant de commencer une mission, mais je suis capable d'apprendre en cours de route ; c'est un fait qui m'a rassuré, à la fois pour la fin de ce stage et pour mon début de carrière future.

Chapitre 4

Séparation multicanale par *machine learning*

4.1 Principes et notations du problème et de la solution

À partir de maintenant et jusqu'à la fin de ce rapport, l'indexation des paramètres et variables du problème commencera à partir de 0, en cohérence avec le code Python écrit. Par exemple, une matrice de taille $J + 1$ aura des lignes indexées de 0 à J . Après avoir présenté la solution de séparation soumise à ICASSP par Lauréline et ses encadrants de thèse, je vais exposer mes contributions dans ce cadre.

4.1.1 Modélisation des signaux

Considérons le mélange additif d'une source d'intérêt (locuteur cible), et d'un ensemble de perturbations composé de J sources interférentes (locuteurs concurrents) et d'un bruit diffus. Les sources non diffuses sont considérées comme ponctuelles. Ce mélange est enregistré avec une antenne de $N = 4$ microphones (format ambisonique).

Dans le domaine transformé, le vecteur $\mathbf{x}(t, f)$ des N canaux du mélange peut s'écrire :

$$\mathbf{x}(t, f) = \mathbf{s}(t, f) + \mathbf{n}(t, f)$$

où t est une trame temporelle, f une fréquence, $\mathbf{s}(t, f)$ l'image spatiale des N canaux de la source d'intérêt, et $\mathbf{n}(t, f)$ est l'image spatiale des N canaux de la perturbation, i.e. la somme des images spatiales des N canaux des J sources interférentes et du bruit diffus. Nous traitons le cas sur déterminé : $J + 1 < N$.

Dans ce cadre, le but de la séparation de sources revient à obtenir une approximation de $\mathbf{s}(t, f)$ à partir du mélange $\mathbf{x}(t, f)$. Il s'agit donc du problème classique de séparation de sources nommé « *Cocktail party problem* ».



FIGURE 4.1 – Cocktail party : mélange additif de locutions concurrentes en environnement bruyant

4.1.2 Description du filtrage et de la formation de voie

En notant H la transconjugaison, nous voulons donc construire un filtre multicanal $\mathbf{w}(f)$ et l'appliquer au mélange $\mathbf{x}(t, f)$, de sorte que le signal filtré

$$y(t, f) = \mathbf{w}(f)^H \mathbf{x}(t, f)$$

s'approche le plus possible de $\mathbf{s}(t, f)$ (au sens d'une métrique que nous définirons ultérieurement).

Un tel filtrage est donc :

- spatial, car multicanal, impliquant les N canaux ambisoniques des signaux considérés ;
- linéaire, car obtenu par produit scalaire hermitien.

Par définition, $\mathbf{w}(f)$ est donc un *beamformer*, au sens le plus général du terme. La focalisation ou formation de voie, appelée *beamforming* en anglais, désigne donc le filtrage par *beamformer*, que nous traduirons par « focaliseur » dans la suite. Ce terme fait originellement référence à des filtres basés sur les directions d'arrivées, appelés focaliseurs fixes. Il a ensuite été généralisé à tous les filtres linéaires spatiaux, en particulier à ceux dépendant de leurs données d'entrée (*data-dependent spatial filters* en anglais). Les filtres dépendants des données d'entrées permettent d'obtenir de meilleures performances en raison de leur capacité à prendre en compte les caractéristiques temporelles et fréquentielles des signaux à filtrer. Ces derniers filtres peuvent être adaptatifs ou non, c'est à dire dépendre du temps t ou pas. En pratique, un filtre doit être adaptatif pour être utilisé en temps-réel, mais cela requiert une complexité de calcul beaucoup plus élevée. Pour notre système encore en développement, le choix d'un filtrage non adaptatif a donc été fait. Outre une meilleure vitesse de calcul, cela nous permet d'obtenir un filtre bien plus robuste aux artefacts pouvant survenir à chaque instant t . Dans la suite, nous présentons les focaliseurs utilisés dans le système de séparation qui nous intéresse.

4.1.2.1 Focaliseur anéchoïque ou *beamformer* pleine bande

En cohérence avec la description du format ambisonique, notons $\mathbf{d}_{\theta, \phi}$ le vecteur directionnel pointant vers une source quelconque localisée par ses coordonnées (θ, ϕ) . Ainsi, pour chaque source i (0 pour la source d'intérêt, 1 à J pour les sources interférentes), on définit le focaliseur anéchoïque pointant vers la source i et annulant les sons venant des J autres directions d'arrivée par :

$$\mathbf{b}_i = [\mathbf{d}_{\theta_0, \phi_0}, \dots, \mathbf{d}_{\theta_J, \phi_J}]^\dagger \mathbf{u}_i$$

† désignant la pseudo-inversion et \mathbf{u}_i étant le vecteur dont le coefficient i vaut 1 et dont tous les autres sont nuls. Ce focaliseur est dit *anéchoïque* car il ne prend en compte qu'une information spatiale (la localisation de la source pointée) : il est indépendant des fréquences. Or, en présence de réverbération, le filtre modélisant le mélange dépendant de la fréquence, donc le filtre de séparation aussi. Ainsi, l'approximation de $\mathbf{s}(t, f)$ par l'image du mélange enregistré par le focaliseur anéchoïque, $y_S(t, f) = \mathbf{b}_0^H \mathbf{x}(t, f)$, se révèle insuffisante pour notre objectif de rehaussement.

4.1.2.2 Une approximation de rang 1 du filtre de Wiener GEVD

Il nous faut donc utiliser un filtre dépendant de la fréquence, plus performant en environnement réverbérant. Dans ce cadre (mélange additif entre une source d'intérêt réverbérée et des sources interférentes), le filtre linéaire optimal d'un point de vue statistique est un filtre de Wiener multicanal par bande de fréquence (idéal).

En pratique, diverses variantes sont implémentées selon les usages. Dans l'état de l'art et notamment dans [24], plusieurs variantes de rang 1 de ce focaliseur sont étudiées pour faire de la reconnaissance, et le plus prometteur (en date de soumission de cet article) est basée sur la décomposition en valeur propre généralisée de ce filtre : on parle de *Generalized Eigenvalue Decomposition* en anglais, et on nommera donc ce focaliseur \mathbf{w}_{GEVD} .

Notons respectivement $\Phi_{ss}(f)$, $\Phi_{nn}(f)$ les matrices de covariances de la source d'intérêt et de la perturbation et $\Phi_{ss-r1}(f)$ l'approximation de rang 1 de $\Phi_{ss}(f)$ utilisée, toutes trois de taille $N \times N$. L'expression de cette dernière, donnée dans [17] et démontrée dans [24], fait intervenir les éléments propres de $\Phi_{nn}(f)^{-1} \Phi_{ss}(f)$. Le focaliseur GEVD est alors :

$$\mathbf{w}_{GEVD}(f) = [\Phi_{ss-r1}(f) + \Phi_{nn}(f)]^{-1} \Phi_{ss-r1}(f) \mathbf{u}_0$$

où \mathbf{u}_0 est défini dans le sous-paragraphe précédent. Il suffit donc d'avoir les matrices de covariances de nos sources pour implémenter ce filtre. En pratique, estimer la matrice de covariance de la source cible nécessite

d'avoir déjà une estimation $\tilde{\mathbf{s}}(t, f)$ de l'image spectrale de cette source sur toutes les T trames temporelle de la locution analysée. On a alors :

$$\Phi_{\mathbf{ss}}(f) = \frac{1}{T} \sum_{t=0}^{T-1} \tilde{\mathbf{s}}(t, f) \tilde{\mathbf{s}}^H(t, f)$$

On procède de même pour $\Phi_{\mathbf{nn}}(f)$. Finalement, pour l'implémentation du focaliseur GEVD, la question cruciale est l'obtention de ces estimations $\tilde{\mathbf{s}}(t, f)$ et $\tilde{\mathbf{n}}(t, f)$.

4.1.2.3 Solution globale proposée

Parmi les diverses solutions exposées dans [10], celle retenue est d'obtenir $\tilde{\mathbf{s}}(t, f)$ en appliquant au mélange un masque temps-fréquence monocanal $M_S(t, f)$ estimé par des réseaux de neurones : $\tilde{\mathbf{s}}(t, f) = M_S(t, f)\mathbf{x}(t, f)$. On peut ensuite en déduire $\tilde{\mathbf{n}}(t, f)$ grâce au masque $M_N(t, f) = 1 - M_S(t, f)$, et ainsi remonter aux matrices de covariances et au focaliseur GEVD comme décrit précédemment.

Les réseaux de neurones constituent un outil pour faire de l'apprentissage automatique (*machine learning* en anglais). Il s'agit, pour un système informatique (ici, le réseau de neurones), d'acquérir de la connaissance suite à la confrontation à différents exemples ou observations connu(e)s, pour apprendre une correspondance entre des données d'entrées et des sorties arbitraires associées. Avec cette connaissance acquise, le système doit ensuite être capable de généraliser pendant une phase de prédiction, c'est-à-dire associer une sortie cohérente à une observation inconnue donnée en entrée. On parle de *deep learning* (apprentissage profond) lorsque l'on utilise plusieurs systèmes les uns à la suite des autres, et que la sortie d'un système devient l'entrée d'un autre. Ici, il s'agit de « réseau de neurones profond », car différentes couches de neurones sont utilisées séquentiellement. Un résumé plus formel de ces notions, ainsi qu'une explication plus précise du concept de profondeur d'un réseau de neurones, sont disponibles dans la partie 2.4.3 de [14].

La construction du réseau de neurones calculant le masque $M_S(t, f)$ repose sur le choix de la sortie que l'on va lui apprendre à prédire, des ses entrées, et de son architecture :

- Pour la phase d'entraînement de ce problème de régression, on choisit comme cible le masque de Wiener monocanal $M_{S,Wiener}(t, f) = \frac{s_W(t, f)^2}{s_W(t, f)^2 + n_W(t, f)^2}$, où $s_W(t, f) = |\mathbf{s}_W(t, f)|$ et $n_W(t, f) = |\mathbf{n}_W(t, f)|$ sont respectivement les magnitudes des images spectrales des signaux de voix, c'est-à-dire les valeurs absolues des composantes omnidirectionnelles respectives de $\mathbf{s}(t, f)$ et $\mathbf{n}(t, f)$;
- Pour le choix des canaux d'entrée du réseau, on constate que le fait d'avoir des entrées corrélées avec la sortie désirée favorise l'apprentissage. Ainsi, les expériences précédentes de Lauréline ont montré qu'une séparation performante sur un mélange donné nécessite des informations d'identification des sources. Pour l'instant les canaux d'entrée retenus sont la magnitude $x_W(t, f)$ de la composante omnidirectionnelle du mélange, et les focaliseurs pointant vers les sources $\hat{s}(t, f) = |\mathbf{b}_0^H \mathbf{x}(t, f)|$ et $\hat{n}_i(t, f) = |\mathbf{b}_i^H \mathbf{x}(t, f)|$ pour $i = 1 \dots J$. Dans notre cas, nous ne considérons qu'une source interférente $i = 1$, que nous noterons $\hat{n}(t, f) = |\mathbf{b}_1^H \mathbf{x}(t, f)|$. On devine l'intérêt de ces entrées en considérant les focaliseurs comme des approximations des images spectrales individuelles nécessaires au calcul de la cible, et le mélange comme une information sur l'ensemble de ces images spatiales : $\hat{s}(t, f)$ approxime $s_W(t, f)$, $\hat{n}(t, f)$ approxime $n_W(t, f)$, et $x_W(t, f)^2$ se rapproche de $s_W(t, f)^2 + n_W(t, f)^2$. Soulignons le fait que la donnée de ces focaliseurs nécessitent la connaissance des directions d'arrivées des sources (notées *DoAs*, *directions of arrival*, sur la figure 4.2). Une localisation doit donc être effectuée en amont ;
- L'architecture choisie sera détaillée dans le paragraphe suivant. Elle doit être cohérente avec les entrées et les sorties choisies, et reposera essentiellement sur une couche d'unités récurrentes LSTM (*Long-Short Term Memory*, [15]), en raison de la structure séquentielle des signaux traités.

4.1.3 Architecture du réseau

Les travaux de Lauréline ont mené à l'architecture de réseaux de neurones présentée en figure 4.3. Les canaux d'entrée sont respectivement x_W , $\hat{s}(t, f)$ et $\hat{n}(t, f)$. Les signaux sont traités par le réseau séquence par séquence. Chaque séquence comporte $T = 25$ trames temporelles (*timesteps* dans le code), et chaque trame temporelle contient $nBand = 513$ bandes de fréquences. Pour rappel, les canaux d'entrées sont $\hat{s}(t, f) = |\mathbf{b}_0^H \mathbf{x}(t, f)|$, $\hat{n}(t, f) = |\mathbf{b}_1^H \mathbf{x}(t, f)|$ et $x_W(t, f) = |\mathbf{x}_W(t, f)|$. En concaténant ces $nFeat = 3$ canaux d'entrées de chaque bande de fréquences, on obtient une entrée de dimension (*timesteps* = 25, $nFeat * nBand$ = 1539).

Pour chaque séquence, la sortie du réseau est le masque $M_S(t, f)$ de dimension (*timesteps* = 25, $nBand$ = 513).

Nous allons dans la suite détailler le rôle de chacune des couches utilisées pour ce réseau.

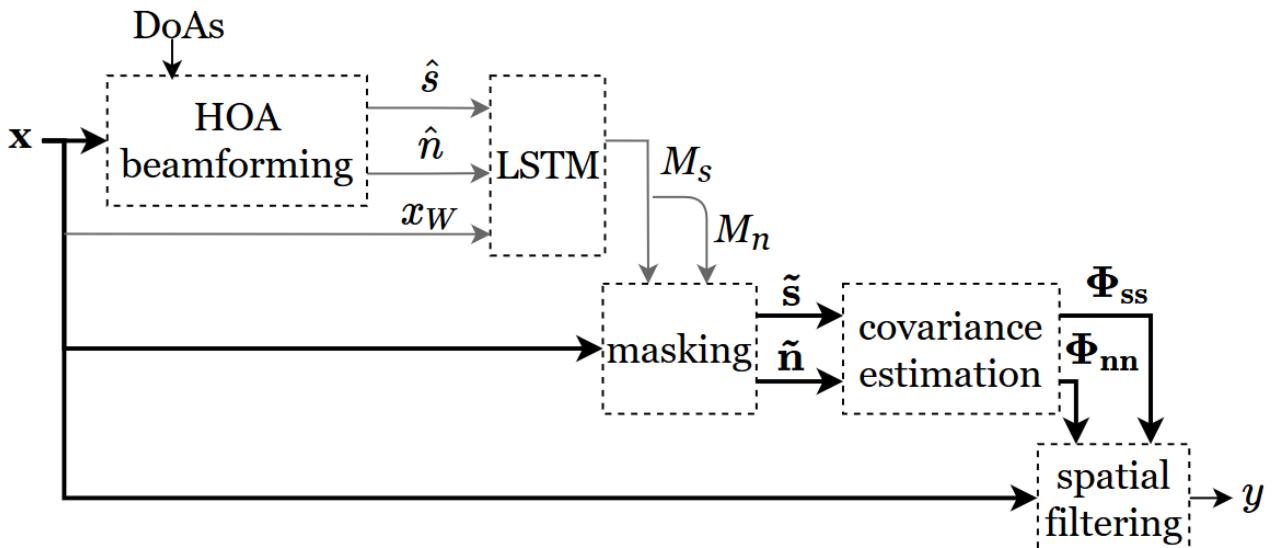


FIGURE 4.2 – Le système de réhaussement proposé par Lauréline et ses encadrants de thèse. Source : [17]

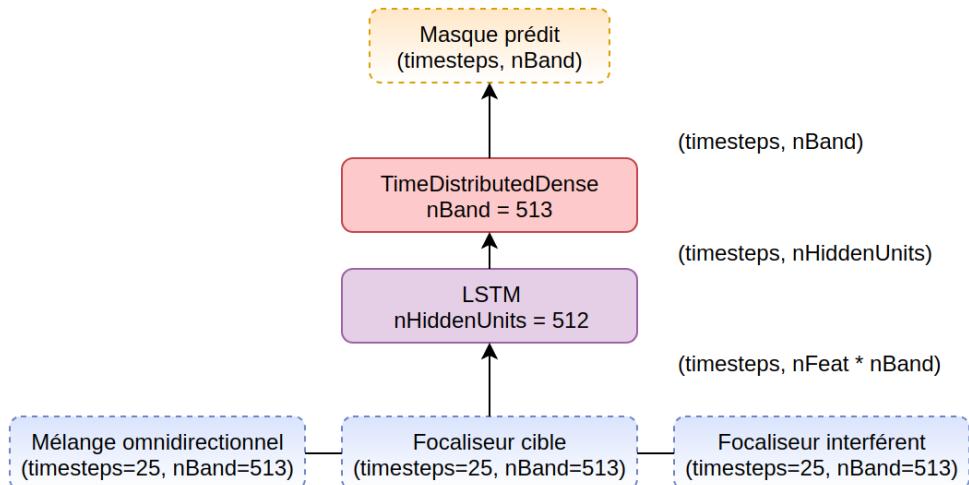


FIGURE 4.3 – Architecture du réseau estimant le masque

4.1.3.1 Couche dense

La couche dense est la plus basique des couches de réseaux de neurones. Il s'agit ici de 513 neurones artificiels caractérisée par la même fonction d'activation, ici la fonction sigmoïde. Quelque soit la taille m d'un vecteur d'entrée de dimension $(1, m)$, la sortie de cette couche aura pour dimension $(1, 513)$. Elle est utilisée en sortie de réseau afin de retrouver la structure en bandes de fréquences du masque prédit.

4.1.3.2 LSTM

Les réseaux LSTM sont des réseaux de neurones récurrents particuliers. Un réseau de neurones récurrent est conçu pour prendre en compte l'aspect séquentiel des vecteurs qui lui sont donnés en entrée (ce qui est très approprié pour le traitement de signaux de parole). Cela se fait en fournissant la sortie d'une unité du réseau à l'instant t en plus de l'entrée de la même unité à l'instant $t + 1$. Un réseau récurrent utilise une transmission de mémoire d'un instant au suivant (voir [15]). Les réseaux LSTM améliorent cette approche en utilisant des portes d'oubli, afin de ne conserver que les informations pertinentes pour le futur.

Ici, nous avons choisi d'utiliser un réseau LSTM de 512 unités cachées pour traiter nos séquences de 25 trames temporelles. Pour chaque séquence, le réseau LSTM va considérer chaque trame comme un état du signal d'entrée. Avec l'implémentation de Keras¹, il peut au choix renvoyer le dernier état (la dernière trame

1. <https://keras.io/layers/recurrent/#lstm>

de la séquence de sortie), ou bien une séquence de 25 trames de dimension 512. C'est cette dernière option qui a été retenue, en cohérence avec le choix d'un système non-adaptatif (expliqué en 4.1.2).

4.1.3.3 Time Distributed Wrapper

Le *Time Distributed Wrapper*² de Keras est un raccourci pour appliquer un même réseau de neurones à chaque trame temporelle qu'il reçoit en entrée (d'où son nom : il « distribue » temporellement un réseau de neurones donné). Nous parlons de couche *TimeDistributedDense* en figure 4.3 car c'est une couche dense que nous distribuons sur chaque trame d'une séquence donnée.

Ici, en sortie du réseau LSTM, nous avons une séquence de 25 trames de dimension 512. Chaque trame est donc représentée par un vecteur de dimension $(1, 512)$. En donnant une telle trame à une couche dense de 513 unités, nous obtiendrions en sortie un vecteur de dimension $(1, 513)$. En distribuant la couche dense sur les 25 trames, nous obtenons donc en sortie du *wrapper* une séquence de dimension $(25, 513)$, ce qui correspond bien aux dimensions attendues pour le masque temps fréquence que nous souhaitons prédire.

4.1.3.4 Hyper-paramètres

Fonctions d'activation : Une fonction d'activation est appliquée à la sortie d'un neurone artificiel. En général, une unique fonction d'activation est utilisée pour une couche donnée du réseau de neurones. Diverses propriétés mathématiques rendent une fonction d'activation plus ou moins intéressante, notamment celles favorisant la convergence lors de la rétro-propagation (différentiabilité, monotonie, etc). En particulier, pour un problème de regression (approximation d'une fonction donnée, ici le masque de Wiener), il est désirable que les fonctions d'activation soient non-linéaires (en effet, un réseau de neurones profond aux fonctions d'activations non-linéaires devient un approximateur de fonction universel au sens décrit dans [4]). Les activations choisies dans notre cas sont classiques : la tangente hyperbolique pour le réseau LSTM, et la sigmoïde pour la couche dense distribuée sur chaque trame de la séquence de sortie du réseau LSTM.

Paramètres d'apprentissage : Les performances du réseau sont calculées via l'erreur quadratique moyenne, appelée MSE pour *Mean Squared Error* en anglais (voir 4.1.4.2). L'optimisation est effectuée grâce à une descente de gradient effectuée selon l'algorithme de Nadam ([7, 22]) avec un taux d'apprentissage initial de 10^{-3} . Elle est appliquée itérativement sur des sous-ensembles de la base d'apprentissage (*batchs*) constitués de 175 exemples. L'application de la descente de gradient à l'ensemble des batchs constitue une « époque » (*epoch*³ en anglais).

Régularisation : Il s'agit de toute les méthodes permettant de prévenir le sur-apprentissage. La méthode de régularisation la plus classique consiste en l'ajout d'un terme de pénalité à la fonction de coût que l'on minimise lors de la rétro-propagation. La régularisation est un concept statistique appliqué au delà de l'apprentissage automatique : dans [20], l'essentiel du concept présenté dans [2] est résumé dans notre cadre. Pour la couche dense du réseau, nous régularisons avec la norme L2, avec un coefficient de 10^{-4} .

Pour le réseau LSTM, la méthode de régularisation utilisée est le *dropout* [21]. Un même coefficient de 0.5 est utilisé pour les entrées et les états récurrents (respectivement *dropout* et *recurrent dropout*) de chaque unité.

Enfin, le sur-apprentissage est également évité en contrôlant les performances du réseau sur un ensemble de validation distinct de l'ensemble d'apprentissage. Lorsque les performances commencent à décroître sur l'ensemble de validation, l'apprentissage est interrompu (*early stopping*).

4.1.3.5 Normalisation

Si l'on entraîne un réseau avec des données n'ayant pas la même échelle, deux problèmes peuvent se poser :

- si le réseau réussit tout de même à trouver une « cohérence » parmi les données qu'il reçoit, l'information apportée par les exemples d'entraînement d'échelles différentes ne sera pas prise en compte de la même manière : le « poids » de chaque exemple est différent, ce qui n'est pas souhaité ;
- lors de la rétro-propagation, les valeurs relativement grandes engendrent de plus forts gradients, ce qui peut saturer les fonctions d'activation et nuire à l'apprentissage.

Pour éviter ces problèmes, nous normalisons statistiquement les données d'entrée juste avant de les fournir au réseau, afin d'avoir une échelle homogène et cohérente avec les activations choisies : il s'agit ici d'obtenir des bases de données de moyenne nulle et de variance égale à 1.

Afin de normaliser de manière cohérente entre l'apprentissage et l'inférence :

- nous calculons les statistiques (moyennes et variances) utilisées pour la normalisation lors de la phase d'apprentissage ;
- nous normalisons ensuite les données de tests avec les statistiques d'entraînement.

2. <https://keras.io/layers/wrappers/#timedistributed>

3. <https://keras.io/getting-started/faq/#what-does-sample-batch-epoch-mean>

4.1.4 Évaluation des modèles

4.1.4.1 Le Word Error Rate (WER)

Nous utilisons ici ce système de séparation dans un cadre de rehaussement de la parole, à des fins de reconnaissance. Rigoureusement, les performances de ce système sont donc à mesurer avec une métrique adaptée à ce but final.

L'unité de mesure couramment utilisée pour la reconnaissance est le *Word-Error-Rate* (WER), taux d'erreur positif comptant le nombre d'insertions I, le nombre de suppressions D et le nombre de substitutions S requises pour que la phrase de référence comportant N mots devienne la phrase transcrive :

$$WER = \frac{S + D + I}{N} \quad (\text{voir [12] et page 28 [14] pour des détails}). \quad \text{Nous utiliserons aussi, de manière interchangeable, l'}accuracy: acc = 1 - WER.$$

4.1.4.2 L'erreur quadratique moyenne (MSE)

Nous souhaitons minimiser le WER i.e. maximiser l'*accuracy* en sortie du filtre GEVD. Cependant, calculer le WER d'un dataset est long :

- Il faut un réseau de neurones entraîné pour effectuer une inférence. Selon les différents paramètres (dont l'*early stopping* [19]), l'entraînement peut prendre entre 30 minutes et une heure ;
- Pour mesurer les performances des inférences, nous utilisions une base de fichiers audio bruts de 20 minutes [9]. Nous calculions le WER sur la base de tests à différentes étapes de la chaîne de traitement : avant tout traitement, après réverbération, après mélange avec les perturbations, après formation de voie, puis après filtrage complet. De plus, pour avoir des éléments de comparaisons, le WER était aussi mesuré sur les masques prédits par le réseau, les masques de Wiener idéaux correspondants, et les filtres GEVD issus respectivement des masques prédits et idéaux. Cela représente donc 8 bases de test de 20 minutes, soit 160 minutes d'enregistrement sur lesquelles il faut lancer la reconnaissance, ce qui dure environ une heure et 45 minutes à chaque fois (en plus de la vingtaine de minutes nécessaires à la génération de ces signaux) ;
- Enfin, pour améliorer encore les performances finales, nous utilisions avant la reconnaissance un logiciel de déréverbération basée sur la méthode *Weighted Prediction Error* (WPE⁴), et ce sur l'ensemble des 160 minutes d'audio générées. Cela rajoutait encore une heure de traitements supplémentaire.

Nous avons donc cherché à utiliser une autre métrique, pour voir si l'on pouvait gagner du temps. Or, le processus d'obtention du focaliseur GEVD à partir du masque prédit par le réseau de neurones est déterministe (masquage du mélange, estimation des matrices de covariances, calcul du filtre). Au cours du stage, nous avons donc eu l'idée d'étudier la corrélation entre les performances de prédiction du réseau de neurones et les performances finales sur la reconnaissance. En effet, supposons que le WER obtenu avec un focaliseur GEVD varie conjointement avec la distance entre le masque idéal et le masque prédit engendrant le dit focaliseur. Alors comparer les performances de prédiction du réseau reviendrait à comparer les WER obtenus après focalisation, et ce sans avoir besoin de lancer (la dé-réverbération et) la reconnaissance sur les bases de test, ce qui éviterait un temps de calcul non négligeable ...

Pour vérifier cette hypothèse, nous avons choisi l'erreur quadratique moyenne (*Mean Squared Error*, MSE) entre les masques prédits et idéaux pour mesurer les performances du réseau. En effet, nous utilisons cette métrique pour optimiser la fonction de coût lors de la phase d'apprentissage du réseau⁵. Cette fonction de coût es calculée sur l'ensemble des séquences des masques :

$$MSE = \frac{1}{nSeq} \cdot \frac{1}{timesteps} \cdot \frac{1}{nBand} \sum_{seq=1}^{nSeq} \sum_{t=1}^{timesteps} \sum_{f=1}^{nBand} [M_S(seq, t, f) - M_{S,Wiener}(seq, t, f)]^2$$

4. <http://www.kecl.ntt.co.jp/icl/signal/wpe/>

5. C'est un choix pertinent en première approche pour un réseau de neurones « simple », mais qui n'a pas l'avantage d'être explicitement lié à notre objectif de minimisation du WER.

Une première figure a été obtenue avec les performances des quelques réseaux disponibles à ce stade de la mission. Avec la seule donnée de cette première courbe, nous en avions déduit, malgré l'absence de relation d'ordre stricte pour les meilleurs réseaux, que les réseaux de performances fortement dissemblables en terme de WER l'étaient aussi en terme de MSE.

Ainsi, nous avions décidé de systématiquement calculer la MSE sur l'ensemble des masques prédits, et d'aviser selon le résultat : si la MSE trouvée semblait d'emblée trop élevée, nous considérions qu'une reconnaissance sur cette base était vaine ; si la MSE était comparable ou inférieure à celle associée à notre réseau de référence (0.1213), nous lancions la dé-réverbération, puis la reconnaissance, pour évaluer plus finement le réseau étudié.

À la fin du stage, avec plus de réseaux, j'ai actualisé cette courbe avec les données des expériences suivantes, ce qui a donné la figure 4.4. Celle-ci confirme que ce critère était très approximatif ...

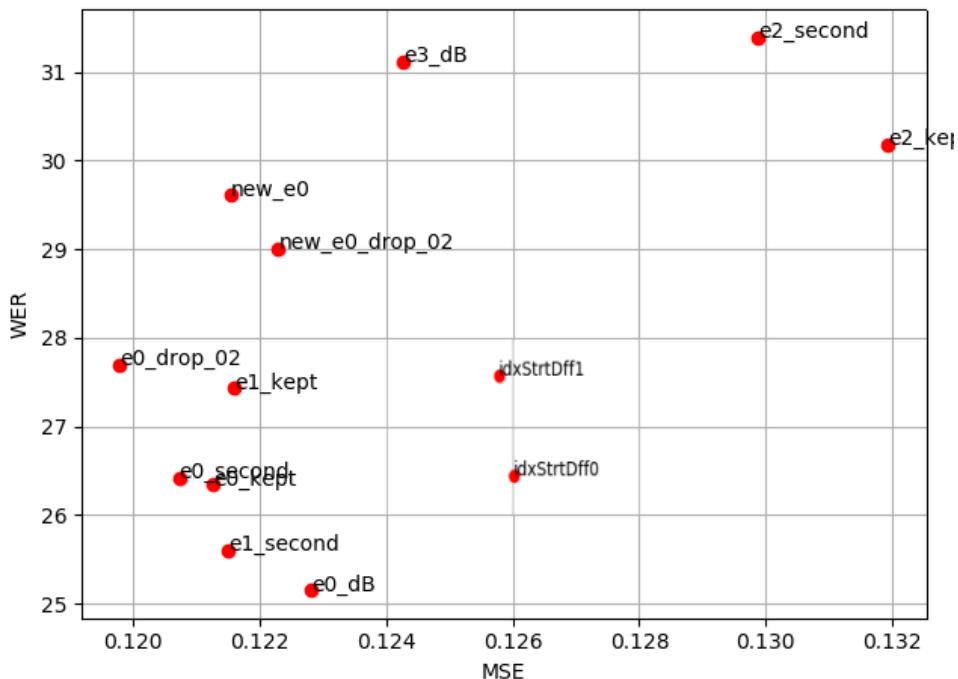


FIGURE 4.4 – WER en fonction de la MSE avec plus de réseaux différents

Néanmoins, ne pas lancer la chaîne complète a permis de gagner du temps, et ainsi tester plus d'idées, pour obtenir un début d'intuition sur des voies d'améliorations plus diverses.

4.2 Réécriture du programme

4.2.1 Organisation générale du système

Décrivons le protocole expérimental suivi pour effectuer une telle séparation de sources.

Lors de la description de chaque étape, nous détaillerons les contraintes que nous avons à respecter.

4.2.1.1 Entraînement et validation

Selection des signaux bruts pour les sources et le bruit : Ici, la principale contrainte est la *diversité* de la base de données d'apprentissage : plus les données sont riches, mieux le réseau sera susceptible de généraliser par la suite. Pour cela, les signaux (bruts, avant tout traitement) utilisés pour générer les locutions et les bruits diffus doivent être les plus divers possibles.

Pour les signaux de voix utilisés à l'entraînement, la base utilisée est un corpus de paroles lues, issue de la revue « Bref »[11]. Elle est constituée de 1801 locutions françaises de 10 secondes, enregistrées dans des fichiers monocanaux. Pour la validation, la base est aussi issue de Bref, mais est constituée de 684 autres signaux. En effet, pour éviter le sur-ajustement, l'entraînement doit être validé avec une base de nature proche de celle d'entraînement, mais différente, pour savoir quand arrêter l'entraînement.

Pour le bruit diffus, nous avons utilisé des enregistrement de bruits de foule issus du site freesound.org, convolus avec la moyenne de la partie diffuse de deux SRIR choisies au hasard. Ces enregistrements ont été sélectionnés au hasard dans des ensembles distincts pour l'entraînement, la validation et le test.

Synthèse de la scène sonore (salle, positions dans la salle, niveaux relatifs) : Pour synthétiser les scènes sonores pour l'entraînement et la validation, nous convoluons les signaux de voix bruts avec des SRIR, pour reproduire l'effet de salle qui affecte la propagation des diverses locutions dans une scène réelle. Ces scènes sonores doivent être synthétisées en respectant les contraintes suivantes :

Salle : Une unique salle pour l'entraînement et une unique autre salle doivent être utilisées pour constituer les bases d'entraînement et de validation. De même que pour le choix des signaux bruts, il faut pouvoir contrôler l'arrêt de l'entraînement en évitant le sur-ajustement. Ainsi, les salles d'entraînement et de validation doivent être comparables, mais caractérisées par des paramètres différents (TR_{60} , distance entre les haut-parleurs et le microphone central utilisés pour mesurer les SRIR ...);

SRIR : Au sein d'une même salle, pour favoriser la diversité des données, il nous faut un maximum de positions possibles. De plus, les positions choisies doivent respecter des contraintes géométriques, notamment en terme de distance angulaire, censées correspondre au cas réel représenté ;

Niveaux relatifs : Ces SRIR sont normalisées de manière à contrôler également les niveaux de puissance relatifs entre les locuteurs concurrents, et entre le locuteur d'intérêt et le bruit diffus. Ces niveaux sont respectivement caractérisés par le SIR (*Signal-to-Interference Ratio*, rapport signal à interférences) et le SNR (*Signal-to-Noise Ratio*, rapport signal à bruit).

Échantillonnage et analyse fréquentielle : Les signaux de voix n'étant pas stationnaires, l'analyse de Fourier doit se faire avec la Transformée de Fourier à Court Terme (TFCT, voir 2.3 de [14] pour détails), sous contrainte d'utiliser une fenêtre d'analyse et de synthèse assurant une reconstruction correcte malgré les artefacts aux interfaces entre les trames des signaux.

Imprécision des mesures : Pour obtenir un système robuste, il faut pouvoir prendre en compte la précision des mesures dans les résultats. Notamment, la définition de l'écart angulaire entre les SRIR se fait aléatoirement, sous contrainte d'une valeur minimale, maximale, ou exacte à respecter. Un paramètre de tolérance d'erreur vient caractériser cette sélection. De même, une erreur moyenne de mesure des directions d'arrivées est autorisable.

Pour paramétrier l'apprentissage, il nous faut maintenant :

- Préparer les données d'entrée (les normaliser, sauvegarder les moyennes et écarts-types d'entraînement, regrouper les trames par séquences qui se chevauchent) ;
- Préparer les masques cibles pour chaque entrée ;
- Définir l'architecture du réseau en conséquence ;
- Configurer l'apprentissage (fonction de coût, algorithme de descente de gradient et son taux d'apprentissage, taille de batch ...) ;
- Régulariser pour éviter le sur-ajustement (avec le *dropout* et le *early-stopping*) .

4.2.1.2 Test

Le protocole appliqué durant la phase de test est très similaire à celui de la phase d'apprentissage ; ici, nous insisterons donc sur les nuances entre ces différentes phases.

Synthèse de la base de test : Les scènes sonores pour le test sont synthétisées avec la même méthodologie que pour celles d'apprentissage et de validation, sauf que :

- Les locutions utilisées pour les sources sont issues du corpus Ester [9]. En effet, pour éviter tout biais scientifique sur la crédibilité des résultats, il faut absolument effectuer le test sur une base de données strictement différente des bases d'apprentissage et de validation. Le corpus Bref était donc ici exclu ;
- Dans le même sens, la salle de test est caractérisée par des paramètres strictement différents de ceux des salles d'apprentissage, qui font de cette salle la moins favorable à de bons résultats (notamment à cause de très fortes premières réflexions).

Prédiction par un réseau entraîné au préalable : comme évoqué en 4.1.3.5, les statistiques d'apprentissage sont utilisées pour le test. Ensuite, après inférence des masques, nous pouvons en déduire les matrices de covariance, puis les filtres multicanaux désirés.

4.2.1.3 Évaluation du réseau :

Comme évoqué en 4.1.4.2, on effectue ensuite le calcul de MSE sur la base de test, et selon le résultat obtenu, on conclut l'étude ou bien on prolonge à chaîne complète avec dé-réverbération et reconnaissance.

4.2.2 Nouvelle implémentation : *Data Generators* et fichiers HDF5

En pratique, on utilise deux outils pour une implémentation optimale.

Les fichiers HDF5 : Les fichiers HDF5 (*Hierarchical Data Format 5*)⁶ sont conçus pour stocker et organiser des grandes quantités de données, avec un système d'arborescence similaire aux dictionnaires. Ils nous permettent d'aller y chercher les données nécessaires sans ouvrir toute la base d'entraînement dans Python⁷ d'un seul coup, ce qui est parfois ingérable pour la mémoire vive de l'ordinateur utilisé ;

Le *fit generator* : il s'agit d'une méthode de Keras pour effectuer l'apprentissage. Contrairement à la méthode classique (*fit*) qui demande toute la base d'entraînement d'un coup (même danger pour la mémoire vive), le *fit generator* prend et charge les exemples d'apprentissage batch par batch.

Chaque batch est généré et chargé par des méthodes spécifiques définies en tant que surcharges d'une classe particulière de Keras, la classe *Data Generator*. Un autre avantage d'utiliser une classe et de construire un code orienté objet, clair et surtout modulaire : si l'on veut par exemple changer la normalisation, il suffit de définir une méthode spécifique supplémentaire dans la classe, sans trop modifier le reste du code.

J'ai mis un peu plus de 6 semaines à implémenter une version du système de séparation de Lauréline, respectant à la lettre le protocole détaillé en 4.2.1 et incluant les gains d'efficacité précédents. Ceci étant fait, il a fallu lancer la chaîne de test complète afin de comparer les performances de mon système au sien.

6. <https://support.hdfgroup.org/HDF5/>

7. Une bibliothèque Python permet de gérer directement ces fichiers dans nos scripts : <https://www.h5py.org/>

4.2.3 Modèle de référence : reproduction de la configuration de Lauréline

Dans cette partie, les résultats obtenus avec ma version du système sont confrontés à ceux de Lauréline. Les canaux d'entrées sont donc

$$\hat{s}(t, f) = |\mathbf{b}_0^H \mathbf{x}(t, f)|, \hat{n}(t, f) = |\mathbf{b}_1^H \mathbf{x}(t, f)| \text{ et } x_W(t, f) = |\mathbf{x}_W(t, f)|$$

et le masque à prédire est

$$M_{S, Wiener}(t, f) = \frac{s_W(t, f)^2}{s_W(t, f)^2 + n_W(t, f)^2}$$

avec l'architecture de réseaux présentée en 4.3. Chacun de ces signaux est découpé en séquences de 25 trames temporelles contenant chacune 513 bandes de fréquences : la dimension des matrices Python est donc

$(nSeq, timesteps = 25, nBand = 513)$, où $nSeq$ correspond au nombre de séquences total, et dépend de la longueur des signaux. Pour l'apprentissage, nous utilisons des fichiers de 10 secondes, ce qui nous donne $nSeq = 24$, et pour la phase de test, les fichiers d'environ une minute (chaque fichier n'a pas exactement une longueur de 60 seconde) ont une valeur plus grande de $nSeq$.

Les paramètres de [17] pour l'apprentissage et pour l'inférence ont donc été utilisés, afin d'obtenir une comparaison cohérente dans la mesure du possible.⁸

4.2.3.1 Spectrogrammes de référence

La figure 4.5 affiche le spectrogramme de la source d'intérêt considérée avant tout traitement. Elle est située à une position $(\theta_0, \phi_0) = (70, 0)$, où θ correspond à l'azimuth et ϕ à l'élévation de la source⁹ :

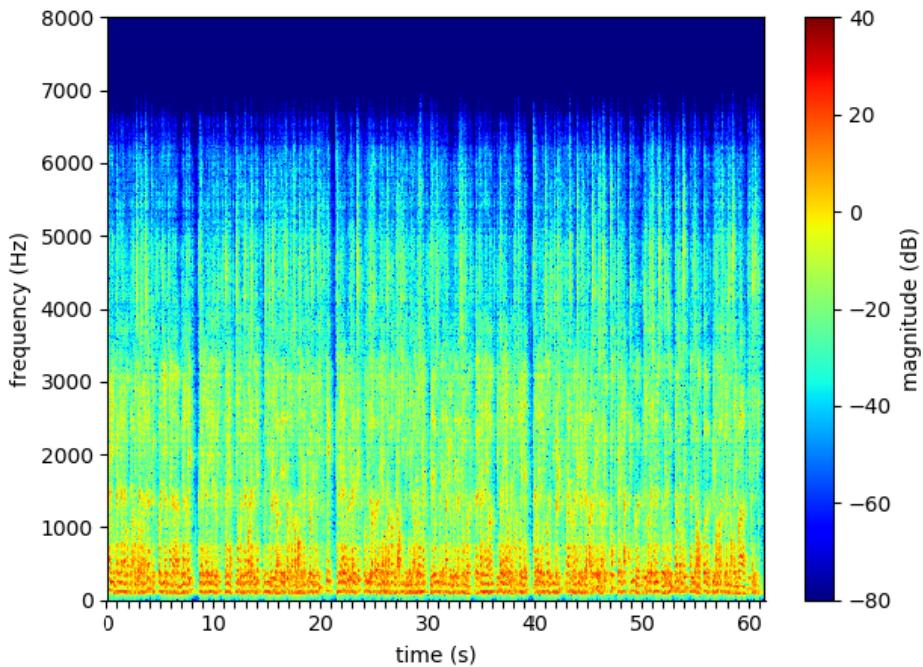


FIGURE 4.5 – Source d'intérêt brute : $s_W(t, f)$

La source interférente étant réglée à une puissance équivalente ($SIR=0$), le spectrogramme de $n_W(t, f)$ est qualitativement similaire et n'est pas tracé ici.

8. Certaines modifications de la structure du code de Lauréline ont inévitablement impliqué une modification des *seeds*, ce qui nous a forcés à effectuer la démarche sur un mélange différent de celui de Lauréline.

9. Dans une salle donnée, la distance au microphone d'enregistrement est fixée et n'est donc pas pertinente ici.

La position de la source interférente est $(\theta_1, \phi_1) = (95, 0)$, ce qui représente une distance angulaire de 25° entre les deux sources. Tous les mélanges respectent exactement cet écart durant l'entraînement, et ont un écart supérieur ou égal pendant la phase de test :

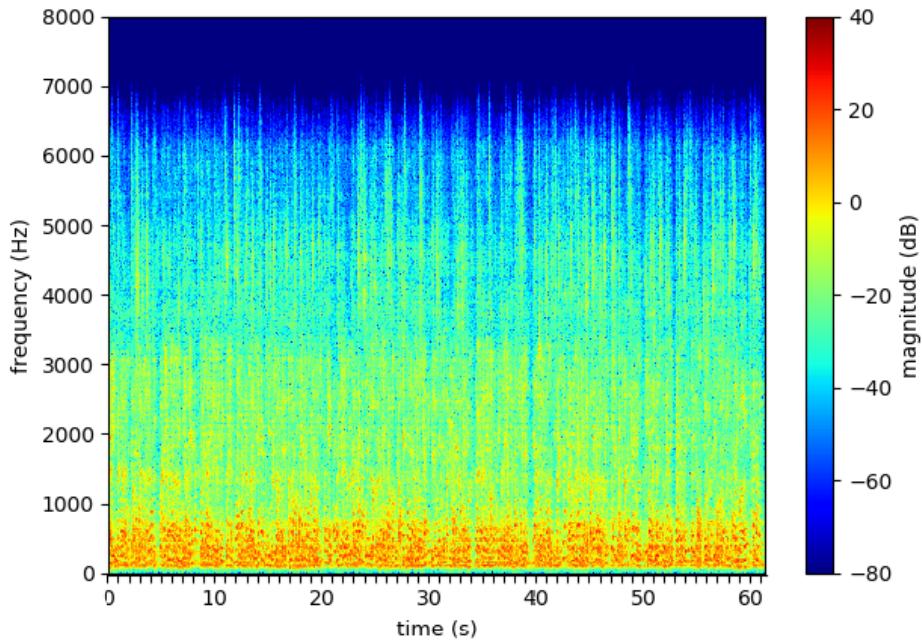


FIGURE 4.6 – Mélange des sources : $x_W(t, f)$

La figure 4.7 correspond à $\hat{s}(t, f) = |\mathbf{b}_0^H \mathbf{x}(t, f)|$, la reconstruction de la source d'intérêt par formation de voie « pleine-bande » à partir du mélange $x_W(t, f)$. Comme énoncé en 4.1.2.1, la séparation n'est pas satisfaisante :

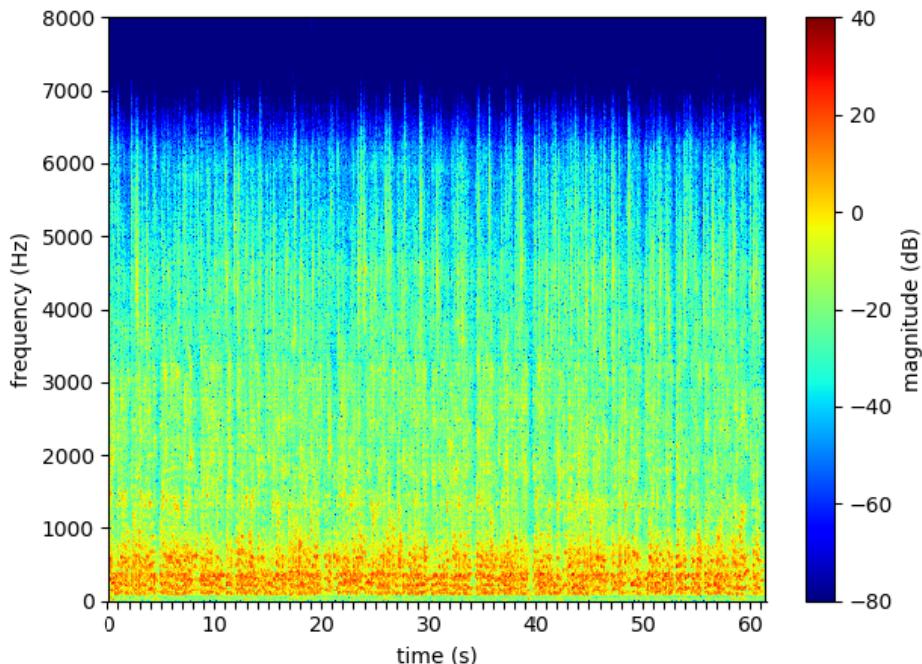


FIGURE 4.7 – Signal d'intérêt reconstruit avec le focaliseur pleine bande correspondant : $\hat{s}(t, f) = |\mathbf{b}_0^H \mathbf{x}(t, f)|$

La figure 4.8 correspond au résultat du masquage du mélange 4.6 par le masque $M_S(t, f)$ prédit par le réseau de neurones, dont le spectrogramme est en 4.9. Ce masquage n'est pas parfait, mais est déjà meilleur que la simple formation de voie pleine-bande précédente :

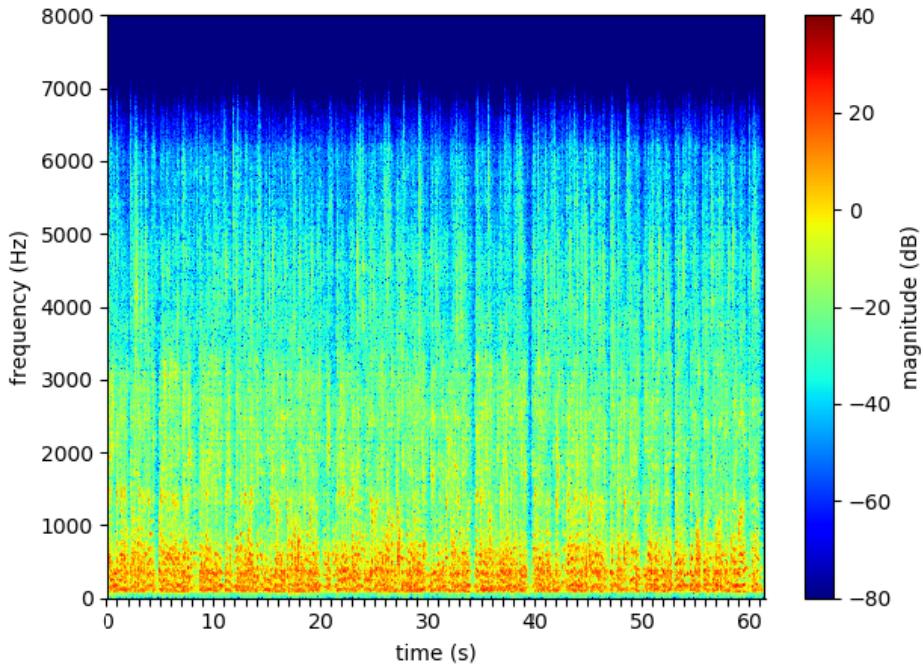


FIGURE 4.8 – Signal d'intérêt reconstruit avec le masque prédit : $\tilde{s}_W(t, f) = M_S(t, f)x_W(t, f)$

L'écart quadratique moyen entre le masque prédit 4.9 et le masque de Wiener idéal 4.11 est de 0.14 pour ce mélange, et on l'obtient 0.1213 en moyennant sur l'ensemble des mélanges de la base de test. La répartition énergétique semble correcte sur l'ensemble des points temps-fréquence considéré, mais le masque obtenu par réseau de neurones est beaucoup moins discriminant localement que le masque idéal.

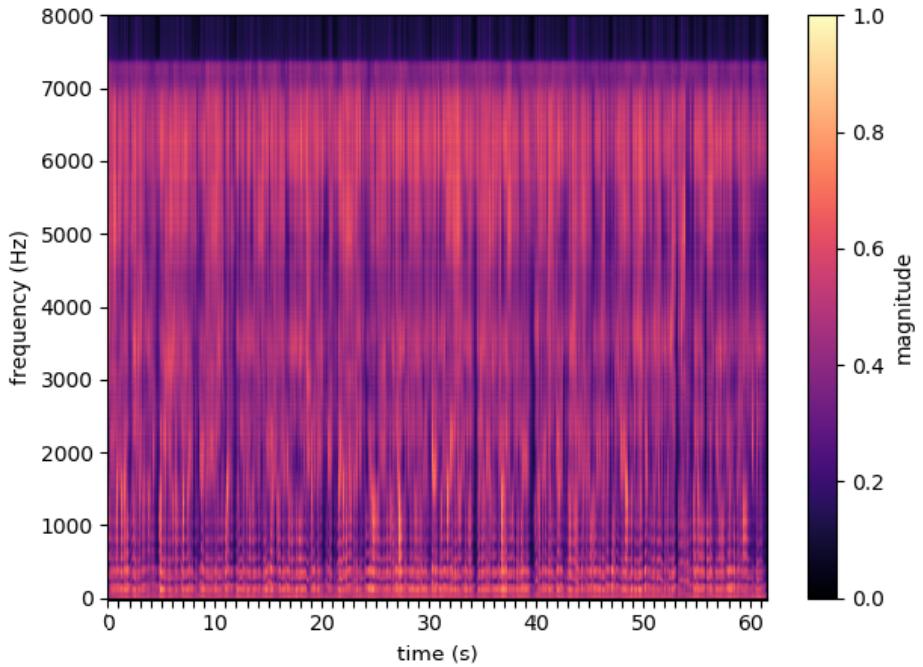


FIGURE 4.9 – Masque prédit par le modèle de référence : $M_S(t, f)$

La figure 4.10 correspond au résultat du masquage du mélange par le masque théorique $M_{S,Wiener}(t, f)$ affiché en 4.11. On constate en comparant les figures 4.8 et 4.10 que le masquage de Wiener est beaucoup plus sélectif que celui que nous avons effectué avec la prédiction du réseau de neurones :

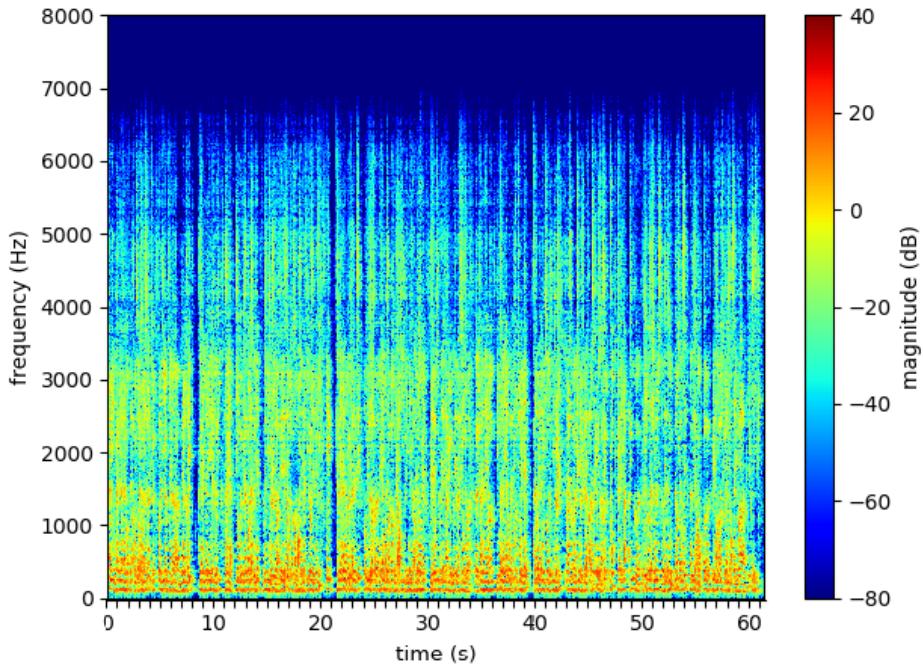


FIGURE 4.10 – Signal d’intérêt reconstruit avec le masque de Wiener : $\tilde{s}_{W,idéal}(t, f) = M_{S,Wiener}(t, f)x_W(t, f)$

Le caractère sélectif de ce dernier est cohérent avec sa définition : le masque de Wiener vaut 1 aux points temps fréquence où seule la source d’intérêt est présente et 0 si celle-ci est absente. Son approximation par réseau de neurones est forcément plus continue vu le critère choisi (MSE sur l’ensemble des points temps-fréquence).

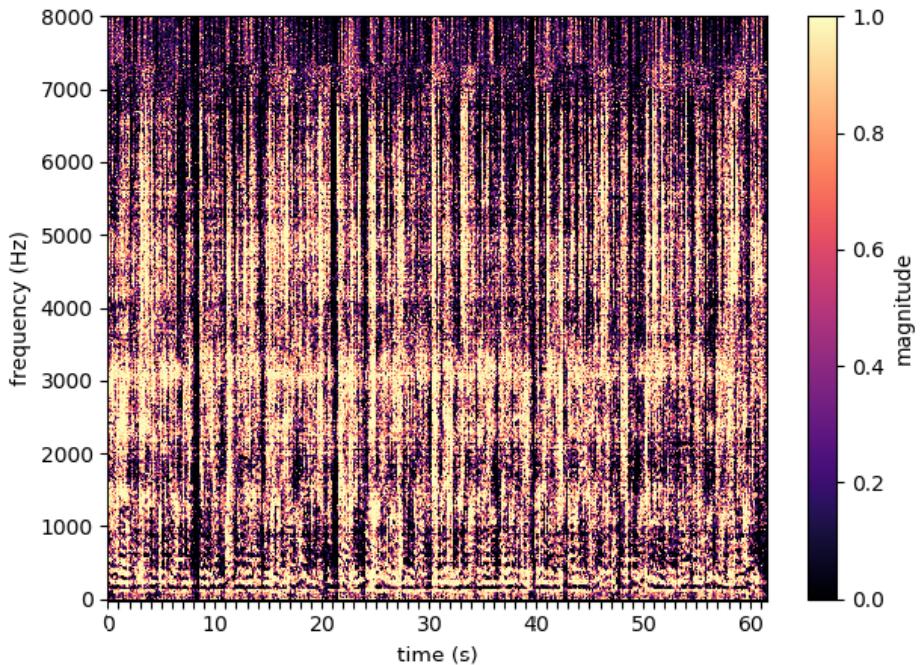


FIGURE 4.11 – Masque de Wiener : $M_{S,Wiener}(t, f)$

La figure 4.12 correspond au résultat du filtrage du mélange par le focaliseur GEVD obtenu grâce au masque prédit 4.9. Ce filtrage est beaucoup plus prononcé qu'en 4.8, mais nous verrons en 4.2.3.2 que cela n'implique pas forcément de meilleures performances en reconnaissance :

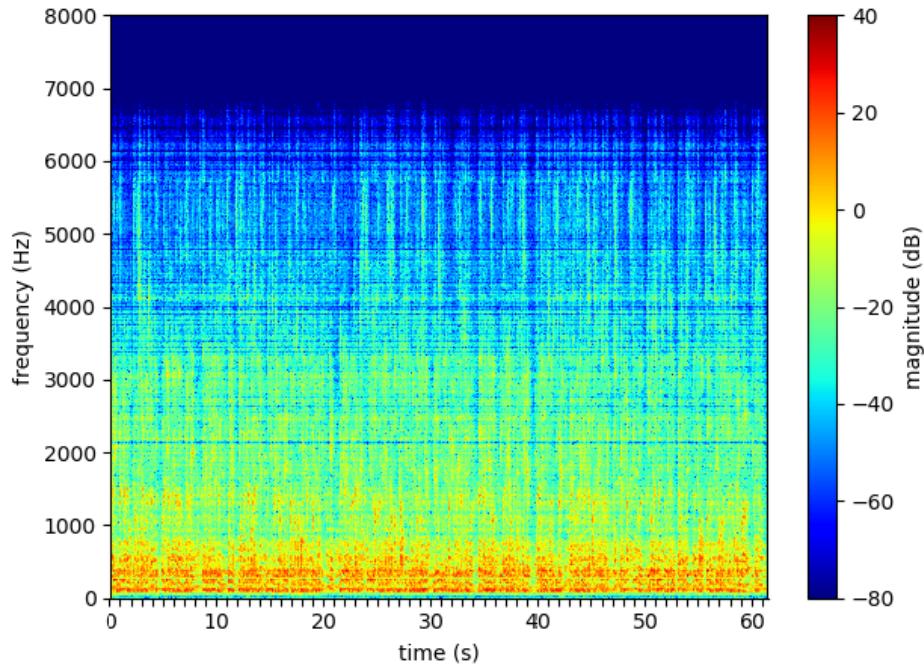


FIGURE 4.12 – Signal d'intérêt reconstruit avec le filtre GEVD prédit : $y(t, f) = \mathbf{w}_{GEVD}(f)^H \mathbf{x}(t, f)$

Voici en 4.13 le résultat du filtrage global obtenu avec le filtre GEVD découlant du masque de Wiener 4.11 :

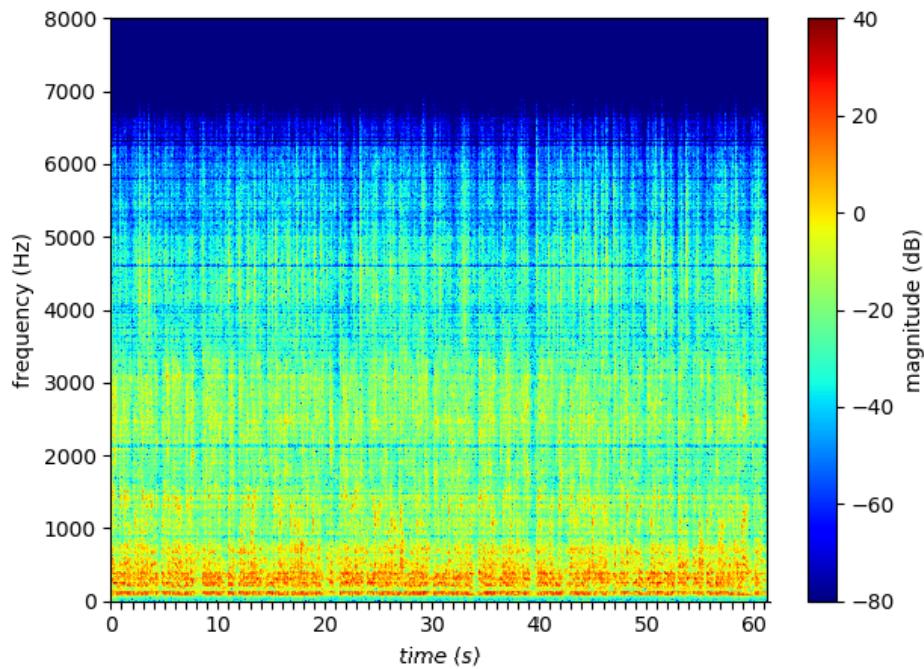


FIGURE 4.13 – Signal d'intérêt reconstruit avec le filtre GEVD idéal : $y_{idéal}(t, f) = \mathbf{w}_{GEVD,idéal}(f)^H \mathbf{x}(t, f)$

4.2.3.2 Résultats de référence

En lançant la dé-réverbération puis la reconnaissance sur chacune des bases de données pertinentes, on obtient les résultats du tableau 4.14. Concernant les résultats de [17], deux mesures sont indisponibles : la mesure de MSE sur les masques prédis avec le système de Lauréline n'est plus pertinente, car la modification des *seeds* nous fait travailler avec une base de test différente ; de plus, j'effectuais le traitement complet sur les signaux réverbérés par les SRIR, ce que Lauréline n'avait pas fait.

	Référence : MSE = 0.1213	Article [17] : MSE indisponible
Source cible brute	92.8	92.6
Source réverbérée	89.7	Indisponible
Mélange capté	10.9	8.3
Focaliseur vers la cible	28.7	24
Masque de Wiener	83.9	83.7
Filtre GEVD idéal	79.5	77
Masque prédit	28.2	39.1
Filtre GEVD prédis	73.7	77.7

FIGURE 4.14 – Comparaison en *accuracy* de mon réseau et de celui de Lauréline

On peut faire plusieurs constats :

- Les résultats de Lauréline et les miens sur les masques et filtres idéaux montrent que parfois, les performances de reconnaissance sur les filtres GEVD ne sont pas meilleures que sur les masques. Cela illustre encore une fois que la méthode, bien qu'ayant fait ses preuves dans la littérature, a la caractéristique de ne pas être directement liée à la métrique de reconnaissance utilisée ;
- Les performances que j'obtiens sur les masques prédis sont clairement moins bons que ceux de Lauréline, sans que cela soit aussi interpellant sur les focaliseurs GEVD.

Le dernier point n'a pas été expliqué avant la fin de mon stage. Cependant, les autres résultats étaient assez cohérent avec ceux de Lauréline pour que je puisse commencer les autres tests, en comparant les résultats à ceux de la première colonne.

Pour simplifier ces comparaisons dans toute la suite du rapport, nous garderons à l'esprit que tous les paramètres sont supposés fixés et identiques à ceux ayant permis d'obtenir cette version de référence du système. Lors de la caractérisation des futurs tests, nous ne citerons que les variables (canaux d'entrées, sorties, hyperparamètres ...) qui diffèrent de nos paramètres de référence.

4.3 Canaux d'entrée du réseau de neurones

Dans toute cette partie sur les entrées du réseaux, nous n'utiliserons que la composante omnidirectionnelle du mélange¹⁰. Par conséquent, sauf contre-indication, lorsque l'on parlera du « mélange », il s'agira de sa composante omnidirectionnelle. De plus, lorsque l'on parlera de focaliseurs, ce sera pour désigner l'ensemble des focaliseurs pointant vers chacune des sources ponctuelles.

4.3.1 La normalisation

Les canaux d'entrées du réseau de Lauréline sont $\hat{s}(t, f)$, $\hat{n}(t, f)$ et $x_W(t, f)$, chacun d'entre eux ayant pour dimension (nSeq, timesteps, nBand). Comme nous l'avons évoqué dans 4.1.3.5, nous normalisons les données avant de les présenter au réseau de neurones. Cette normalisation impacte les performances, et nous avons donc envisagé différentes façons de normaliser, que je vais exposer dans la suite.

Pour cette partie, pour chaque canal d'entrée $I(t, f)$, notons respectivement $m_{I,k}$ et $\sigma_{I,k}$ la moyenne et l'écart-type de $I(t, f)$ sur la k -ième dimension.

De plus, notons $\tilde{I}_k(t, f) = \frac{I(t, f) - m_{I,k}}{\sigma_{I,k}}$ le canal normalisé sur la k -ième dimension. Ces notations seront spécifiées et simplifiées naturellement.

4.3.1.1 Canal par canal

C'est celle utilisée sur le réseau de référence et dans [17]. Il s'agit de calculer les statistiques de chaque canal d'entrée sur l'ensemble des séquences, i.e. sur la première dimension :

$$\tilde{I}_{seq}(t, f) = \frac{I(t, f) - m_{I,seq}(t, f)}{\sigma_{I,seq}(t, f)}$$

Les statistiques $m_{I,seq}(t, f)$ et $\sigma_{I,seq}(t, f)$ sont donc de dimension (timesteps, nBand), étendue à (nSeq, timesteps, nBand) par *broadcast*¹¹. Les canaux d'entrées normalisés deviennent ainsi :

$$\tilde{\hat{s}}_{seq}(t, f) = \frac{\hat{s}(t, f) - m_{S,seq}(t, f)}{\sigma_{S,seq}(t, f)}, \tilde{\hat{n}}_{seq}(t, f) = \frac{\hat{n}(t, f) - m_{N,seq}(t, f)}{\sigma_{N,seq}(t, f)} \text{ et } \tilde{x}_{W,seq}(t, f) = \frac{x_W(t, f) - m_{X,seq}(t, f)}{\sigma_{X,seq}(t, f)}$$

Les résultats obtenus avec cette normalisation sont ceux du tableau 4.14.

4.3.1.2 Unique pour tous les canaux

Ici, il ne s'agit plus de normaliser individuellement chaque canal d'entrée, mais de calculer des statistiques $m_{seq}(t, f)$ et $\sigma_{seq}(t, f)$ communes aux trois canaux. Cela revient à considérer $m_{seq}(t, f)$ comme la moyenne arithmétique de $m_{S,seq}(t, f)$, $m_{N,seq}(t, f)$ et $m_{X,seq}(t, f)$, puis à calculer $\sigma_{seq}(t, f)$ à partir de $m_{seq}(t, f)$ après concaténation par rapport aux bandes de fréquences (dernière dimension).

Ainsi, les canaux d'entrée normalisés deviennent

$$\tilde{\hat{s}}_{seq}(t, f) = \frac{\hat{s}(t, f) - m_{seq}(t, f)}{\sigma_{seq}(t, f)}, \tilde{\hat{n}}_{seq}(t, f) = \frac{\hat{n}(t, f) - m_{seq}(t, f)}{\sigma_{seq}(t, f)} \text{ et } \tilde{x}_{W,seq}(t, f) = \frac{x_W(t, f) - m_{seq}(t, f)}{\sigma_{seq}(t, f)}$$

toujours de dimension (nSeq, timesteps, nBand).

L'objectif est ici de réduire le nombre de paramètres, tout en conservant l'information de la différence de valeur entre chaque canal et la valeur moyenne des 3 canaux.

Les résultats de ce type de normalisation sont consignés dans le tableau 4.15. Les performances en reconnaissance sont donc finalement un peu moins bonnes qu'avec le réseau de référence, mais en contrepartie d'un nombre de paramètres à calculer plus faible. En effet, utiliser les mêmes statistiques pour les 3 canaux réduit la complexité du problème.

L'idée peut être approfondie en moyennant en plus sur une autre dimension, c'est ce que j'ai fait dans la suite.

10. Les expériences de Lauréline semblent montrer que les composantes ambisoniques d'ordre 1 du mélange n'apportent aucune information supplémentaire par rapport à la composante omnidirectionnelle. Il semblerait que l'information de localisation contenue dans l'ordre 1 soit résumée par les focaliseurs.

11. <https://docs.scipy.org/doc/numpy-1.13.0/user/basics.broadcasting.html>

	Normalisation unique : MSE = 0.1216	Référence : MSE = 0.1213
Source cible brute	92.8	92.8
Source réverbérée	90.0	89.7
Mélange capté	10.6	10.9
Focaliseur vers la cible	28.4	28.7
Masque de Wiener	83.9	83.9
Filtre GEVD idéal	79.7	79.5
Masque prédit	26.5	28.2
Filtre GEVD prédicts	72.6	73.7

FIGURE 4.15 – Comparaison en *accuracy* de la normalisation unique à la normalisation canal par canal

Unique pour tous les canaux et moyennage en fréquences En partant donc des canaux normalisés précédemment définis, j'ai tenté de faire en plus un moyennage de $m_{seq}(t, f)$ et $\sigma_{seq}(t, f)$ sur toutes les bandes de fréquences, définissant ainsi les canaux normalisés suivants :

$$\tilde{s}_{seq,freq}(t, f) = \frac{\hat{s}(t, f) - \bar{m}_{seq}(t)}{\bar{\sigma}_{seq}(t)}$$

$$\tilde{n}_{seq,freq}(t, f) = \frac{\hat{n}(t, f) - \bar{m}_{seq}(t)}{\bar{\sigma}_{seq}(t)}$$

et

$$\tilde{x}_{W,seq,freq}(t, f) = \frac{x_W(t, f) - \bar{m}_{seq}(t)}{\bar{\sigma}_{seq}(t)}$$

avec

$$\bar{m}_{seq}(t) = \frac{1}{nBand} \sum_f m_{seq}(t, f) \text{ et } \bar{\sigma}_{seq}(t) = \frac{1}{nBand} \sum_f \sigma_{seq}(t, f)$$

Cette fois, la perte de performance en termes de reconnaissance est plus notable.

	Normalisation unique et en fréquences : MSE = 0.1319	Référence : MSE = 0.1213
Source cible brute	92.8	92.8
Source réverbérée	90.1	89.7
Mélange capté	10.5	10.9
Focaliseur vers la cible	28	28.7
Masque de Wiener	83.9	83.9
Filtre GEVD idéal	79.6	79.5
Masque prédit	17.4	28.2
Filtre GEVD prédicts	69.8	73.7

FIGURE 4.16 – Comparaison en *accuracy* de la normalisation en fréquences à la normalisation canal par canal

4.3.2 Entrées de type rapport signal à bruit (SNR)

L'idée de normaliser par l'énergie était également au programme des tests à effectuer pendant ce stage. Une intuition justifiant cette idée (et sa désignation) est celle des rapports signal sur bruit ou SNR (Signal-to-Noise Ratio) : lorsqu'on y a accès, ce type de rapport permet de quantifier la proportion de signal utile par rapport aux signaux perturbateurs, ce qui se rapproche des masques temps-fréquence que nous cherchons à prédire. Ainsi, nous avons eu l'idée de normaliser les entrées $\hat{s}(t, f)$, $\hat{n}(t, f)$ et $x_W(t, f)$ par $x_W(t, f)$. Deux choses sont alors à souligner :

- Les trois canaux d'entrées normalisés deviennent donc $\frac{\hat{s}(t, f)}{x_W(t, f)}$, $\frac{\hat{n}(t, f)}{x_W(t, f)}$ et $\frac{x_W(t, f)}{x_W(t, f)} = J_N$, où J_N désigne la matrice unité¹² de taille $N \times 1$: ce dernier canal n'apporte donc plus la moindre information au réseau, et sera donc supprimé.
- On a d'abord choisi de diviser par $x_W(t, f)$ plutôt que par les interférences pour avoir une quantité généralisable à une, deux, ou plusieurs sources. De plus, en supposant les sources statistiquement indépendantes (hypothèse classique en séparation de sources), la puissance du mélange est égale à la somme des puissances de chaque sources. Ainsi, les rapports précédents sont bornées entre 0 et 1 en échelle linéaire, et donnent donc des valeurs négatives en décibels.

Des variations de cette idée de base ont été testées, et je vais tâcher de décrire le cheminement qui a mené à ces expérimentations. Dans la suite, notons :

$$SMR(t, f) = \frac{\hat{s}(t, f)}{x_W(t, f)}$$

le rapport signal à mélange (Signal-to-Mixture Ratio) et

$$IMR(t, f) = \frac{\hat{n}(t, f)}{x_W(t, f)}$$

le rapport interférence à mélange (Interference-to-Mixture Ratio).

Lors de l'implémentation, la question de l'échelle d'entrée a commencé à se poser (et sera développée plus longuement dans la suite du rapport). En effet, les SNR sont usuellement étudiés en décibels plutôt qu'en linéaire, en particulier dans les domaines du traitement du signal et de l'acoustique. J'ai donc effectué ces premiers tests avec le canal d'entrée du réseau en décibels, en implémentation la possibilité de choisir entre les deux échelles. Pour chaque modèle testé, l'échelle choisie sera donc précisée.

4.3.2.1 Rapports signaux sur mélange (SMR) et focaliseurs

4.3.2.1.1 Une source, $SMR(t, f)$ seul en décibels : Avec en objectif la vérification de mon code et la pertinence de l'idée, j'ai commencé par tester un modèle avec une seule source. Dans ce cas simple, le mélange se résume à la somme de la source d'intérêt et du bruit diffus (le canal IMR ne peut être calculé ici). Le canal d'entrée, donné en décibels, est donc

$$SMR_{dB}(t, f) = 20 \log_{10} \frac{\hat{s}(t, f)}{x_W(t, f)}$$

On peut voir sur les figures 4.17 et 4.18 que le focaliseur pointant vers la source d'intérêt a un spectrogramme très proche de celui du « mélange ».

12. À ne pas confondre avec les matrices « unitaires ».

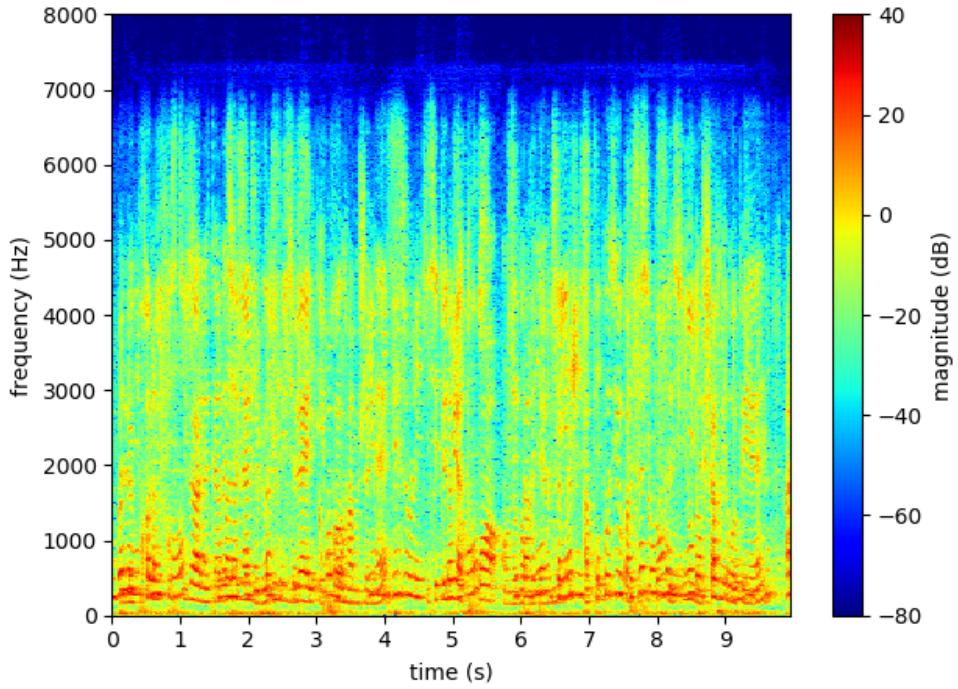


FIGURE 4.17 – « Mélange » pour une source : $x_W(t, f)$

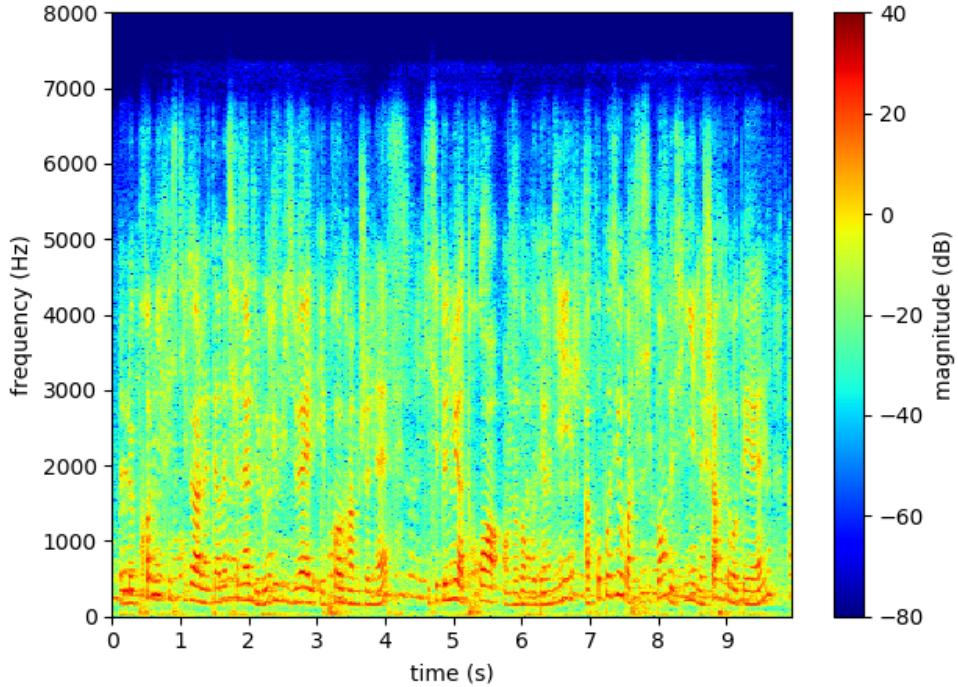


FIGURE 4.18 – Focaliseur pointant vers l’unique source : $\hat{s}(t, f)$

Ici, le SNR (source d’intérêt sur bruit diffus) est réglé à une valeur relativement élevée (20 dB), ce qui explique que la puissance du mélange soit proche de celle de la source d’intérêt. On s’attend à un SMR variant autour de 0 dB, avec une dynamique relativement faible. Or, ce SMR doit être négatif (en décibels) par construction (plus de puissance dans le mélange que dans une de ses composantes isolée)¹³. On s’attend donc à un spectrogramme avec une majorité de points chauds à des magnitudes négatives proches de 0 dB.

13. De fait, il existe des points temps-fréquence au SMR positifs à cause des effets-de-bord aux bornes des trames de la TFCT. Cela a été corrigé artificiellement par un seuillage lors de l’implémentation.

Le spectrogramme en figure 4.19 correspond en effet à cette description et semble confirmer que l'implémentation est correcte. Une inférence avec comme seule entrée le SMR en débibel a donné une MSE de 0.1299.

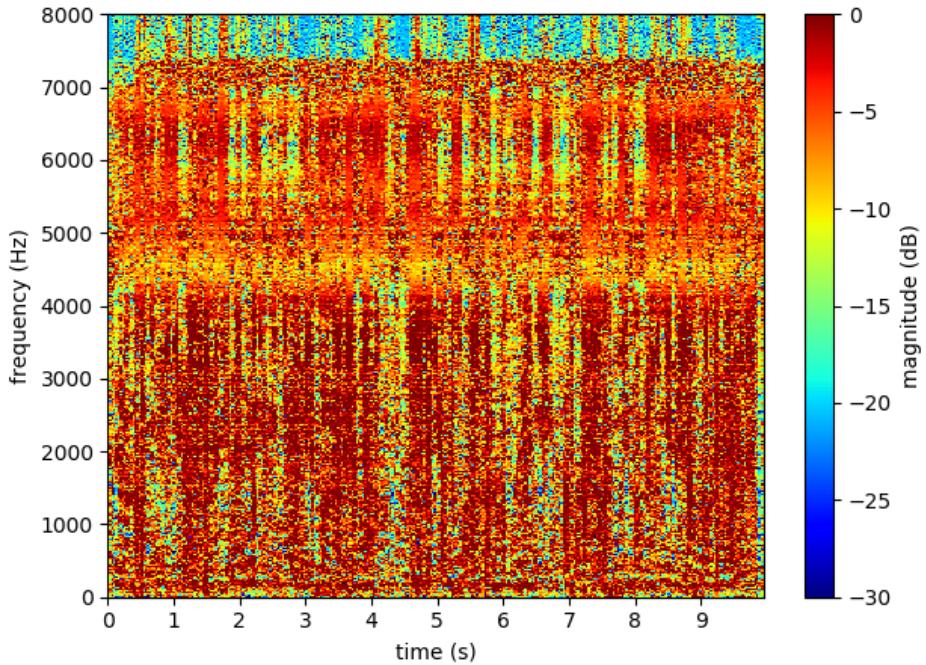


FIGURE 4.19 – SMR de l'unique source : $SMR(t, f) = \frac{\hat{s}(t, f)}{x_W(t, f)}$

4.3.2.1.2 Une source, $SMR(t, f)$ et $\hat{s}(t, f)$ en décibels : Au cours de cette implémentation, Alexandre et moi avons pensé à reprendre une idée des travaux précédents de Lauréline. Dans le cas d'une source, elle avait pour canaux d'entrée le focaliseur vers la source d'intérêt $\hat{s}(t, f)$, donnant des informations d'identification de la cible au réseau, et le mélange, donnant des informations sur la proportion de signal utile et de perturbation. On a donc repris $\hat{s}(t, f)$ en tant que second canal d'entrée, car il permet d'ajouter une estimation plus proche du signal visé que le SMR seul (qui fournit plutôt l'information de proportion signal / mélange).

J'ai donc lancé une inférence avec le modèle d'entrée le SMR et $\hat{s}(t, f)$ en décibels, et obtenu une MSE de 0.1171 sur la base de tests, ce qui est meilleur que la valeur de 0.1299 obtenue avec le SMR seul. Cela semblait donc confirmer notre intuition et nous encourager à conserver le focaliseur en tant que canal d'entrée. Mais ce résultat intéressant est difficile à interpréter davantage. Il est donc temps de généraliser à plusieurs sources, en gardant en tête de comparer les deux échelles.

4.3.2.1.3 Deux sources, $SMR(t, f)$, $IMR(t, f)$, $\hat{s}(t, f)$ et $\hat{n}(t, f)$: En considérant les résultats obtenus pour une seule source, j'ai tenté de généraliser l'approche à deux sources, en implémentant un modèle à reprenant l'idée de la double entrée focaliseur / rapport énergétique : j'ai implémenté un modèle à quatre canaux, les deux focaliseurs $\hat{s}(t, f)$ et $\hat{n}(t, f)$, pointant respectivement vers la source d'intérêt et la source concurrente, le SMR, ainsi que le rapport interférence à mélange, défini de manière analogue au SMR par

$$IMR(t, f) = \frac{\hat{n}(t, f)}{x_W(t, f)}$$

J'ai d'abord lancé une inférence avec ce modèle en échelle linéaire, on obtient une MSE de 0.1228, contre 0.1213 pour la référence avec les canaux d'entrées de Lauréline. L'écart n'étant pas explicitement significatif, j'ai relancé l'inférence avec l'échelle en décibels pour comparaison. La MSE vaut cette fois 0.1287, ce qui est moins prometteur. J'ai donc lancé le processus de dé-réverbération et de reconnaissance sur la base inférée avec le modèle linéaire, ce qui a donné les résultats suivants :

On constate que les performances sont bien en dessous des résultats de référence. Une hypothèse pour expliquer ces chiffres décevants serait l'excès de données : en voulant favoriser l'apprentissage avec le plus

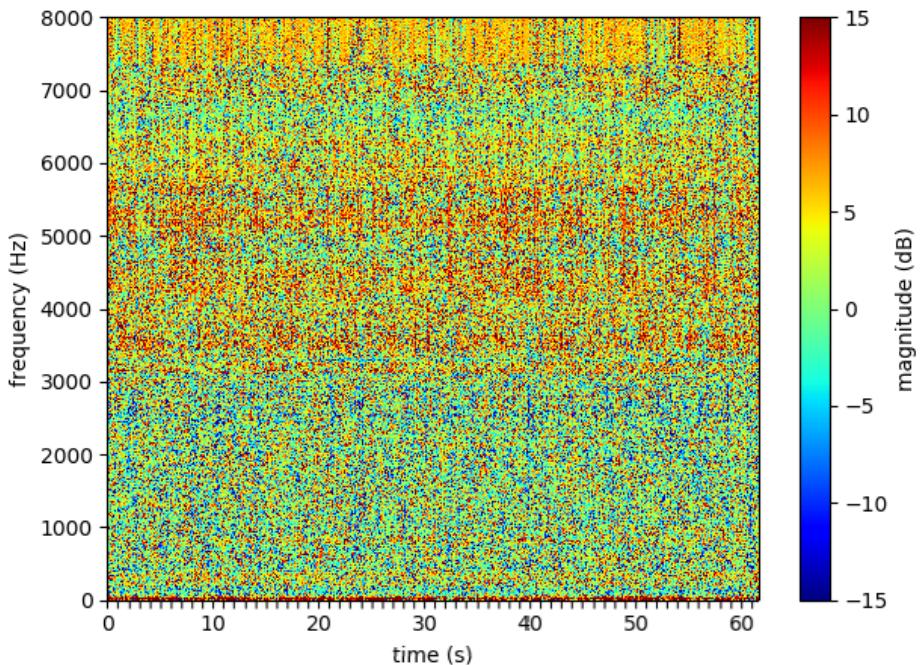
	Rapports énergétiques en entrée : MSE = 0.1238	Référence : MSE = 0.1213
Source cible brute	92.9	92.8
Source réverbérée	90.2	89.7
Mélange capté	10.8	10.9
Focaliseur vers la cible	28.0	28.7
Masque de Wiener	84.0	83.9
Filtre GEVD idéal	79.7	79.5
Masque prédit	21.6	28.2
Filtre GEVD prédits	68.9	73.7

FIGURE 4.20 – *Accuracies* du réseau d’entrées $SMR(t, f)$, $IMR(t, f)$, $\hat{s}(t, f)$ et $\hat{n}(t, f)$ et du réseau de référence

d’information possible, peut-être a-t-on noyé le réseau ? Un système d’entrées moins lourd, avec par exemple $SMR(t, f)$ et $\hat{s}(t, f)$ comme uniques canaux, a été envisagé. Cependant le spectrogramme de $SMR(t, f)$ semblait beaucoup trop bruité pour être porteur d’information. Cela peut s’expliquer par plusieurs facteurs liés au rapport :

- Les signaux de voix étant par nature non-stationnaires, le rapport de deux signaux avec de telles dynamiques est assez intuitivement instable ;
- Les sources n’occupent *a priori* pas les mêmes points temps-fréquence, d’où des interférences difficiles à anticiper, et potentiellement destructives ;
- Les points temps-fréquence où le mélange (au dénominateur) s’annulent donnent lieu à d’importantes divergences du rapport.

Nous avons tout de même tenté de modifier l’expression du rapport énergétique, mais les diverses expressions ont toutes eu le point commun de donner un spectrogramme inutilisable car trop bruité.



$$\text{FIGURE 4.21 – SMR pour deux sources : } SMR(t, f) = \frac{\hat{s}(t, f)}{x_W(t, f)}$$

En cherchant à comprendre la nature de ces résultats, nous avons émis une hypothèse : les canaux d’entrées du réseaux étant donnés en amplitudes au réseau pour prédire des masques en puissance, il y aurait potentiellement

un manque à gagner en utilisant la même échelle pour les entrées et la sortie du réseau. C'est ce que nous avons tenté d'approfondir dans la partie suivante.

4.3.3 Échelle des canaux d'entrée (deux sources)

Par définition de $WMS(t, f)$, nous tentons de prédire des masques dont les valeurs sont supposées comprises dans $[0, 1]$. Ainsi, en appliquant aux masques prédicts des fonctions croissantes sur cette intervalle, on préserve les relations d'ordres de puissances des différents points temps-fréquence, tout en modifiant la dynamique des variations de niveau. Par extension de ce principe et des idées précédentes, nous avons souhaité tester des données au réseau de neurones des entrées quadratiques et logarithmiques pour comparaison à l'échelle linéaire.

4.3.3.1 Entrées quadratiques

Pour les canaux d'entrée au carré, les résultats ont été mauvais :

- Pour les entrées de type SNR, il semblerait que le carré amplifie les problèmes constatés dans la partie précédente. Le SMR au carré est aussi bruité que la figure 4.21 ;
- Nous avons donc tenté d'appliquer cette idée aux canaux originaux de Lauréline : $\hat{s}(t, f)$, $\hat{n}(t, f)$ et $x_W(t, f)$. La MSE obtenue étant de 0.1335 (le plus grand écart trouvé), nous n'avons pas poursuivi.

4.3.3.2 Entrées logarithmiques (modèle de référence)

Encouragé par les résultats décevants de l'échelle quadratique et des canaux d'entrées type rapports énergétiques, nous avons voulu tester la modification de dynamique inverse, en utilisant les entrées initiales du réseau en décibel.

J'ai trouvé une MSE de 0.1228, ce qui fut assez encourageant pour que je lance la dé-réverbération puis la reconnaissance. Les résultats finaux sont les suivants :

	Canaux d'entrées en décibels : MSE = 0.1228	Référence (linéaire) : MSE = 0.1213
Source cible brute	92.8	92.8
Source réverbérée	89.8	89.7
Mélange capté	11.1	10.9
Focaliseur vers la cible	Indisponible	28.7
Masque de Wiener	84.2	83.9
Filtre GEVD idéal	79.5	79.5
Masque prédict	22.6	28.2
Filtre GEVD prédicts	74.9	73.7

FIGURE 4.22 – *Accuracies* du réseau de référence avec les entrées en décibels et linéaires

Il s'agit du meilleur score d'*accuracy* obtenu durant toute la durée du stage, et également du seul strictement supérieur à celui de référence (quoique toujours inférieur au score soumis dans [17] par Lauréline). Il semblerait donc qu'en effet, présenter les canaux d'entrée au réseau avec une échelle logarithmique soit favorable dans notre cas, et que les problèmes précédents soient plus liés à l'instabilité des rapports énergétiques qu'à un problème d'échelle.

La mesure indisponible est due à un problème lors de l'écriture des fichiers WAV correspondants aux canaux d'entrée. Je n'ai pas débugué assez tôt ce problème pour refaire la mesure.

4.4 Optimisation de l'apprentissage

Dans toute cette partie, nous utilisons de nouveau les canaux d'entrée initiaux du réseau de Lauréline :

$$\hat{s}(t, f) = |\mathbf{b}_0^H \mathbf{x}(t, f)|, \hat{n}(t, f) = |\mathbf{b}_1^H \mathbf{x}(t, f)| \text{ et } x_W(t, f) = |\mathbf{x}_W(t, f)|$$

4.4.1 Apprentissage pour le réseau de référence

Le masque à prédire est toujours $M_{S, Wiener}(seq, t, f) = \frac{s_W(seq, t, f)^2}{s_W(seq, t, f)^2 + n_W(seq, t, f)^2}$. Comme les canaux d'entrées, ce masque a pour dimension (nSeq, timesteps, nBand), où chacune des nSeq séquences comporte $timesteps = 25$ trames temporelles comportant $nBand = 513$ bandes de fréquence (nSeq dépendant donc de la longueur des signaux). La figure 4.23 illustre la superposition des différentes séquences : les différentes séquences successives sont colorées respectivement en vert, bleu, rouge et violet. Durant tout le stage, nous avons travaillé avec 12 trames temporelles superposées entre deux séquences « consécutives » .

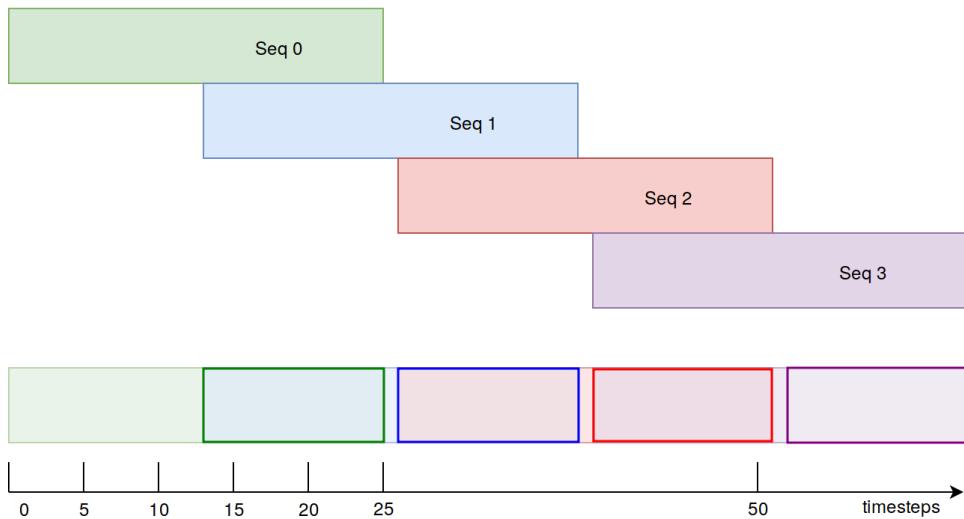


FIGURE 4.23 – Les séquences prédites pour le masque se superposent et sont redondantes.

Lors de l'apprentissage, la distance entre le masque idéal $M_{S, Wiener}$ et le masque prédit M_S est calculée sur l'ensemble des séquences et des points temps-fréquence par la MSE :

$$MSE = \frac{1}{nSeq} \cdot \frac{1}{timesteps} \cdot \frac{1}{nBand} \sum_{seq=1}^{nSeq} \sum_{t=1}^{timesteps} \sum_{f=1}^{nBand} [M_S(seq, t, f) - M_{S, Wiener}(seq, t, f)]^2$$

Cette métrique donne le même poids à chaque trame temporelle. Or, le réseau prédit le masque en exploitant la structure temporelle des données grâce aux unités LSTM. Cela implique que les portes qui activent la mémoire interne des unités s'adaptent aux données. La prédiction étant causale, cette adaptation n'est *a priori* pas correcte sur les premières trames. Ainsi, la métrique est pénalisée par les premières trames non initialisées, ce qui ne semble pas optimal.

D'où l'idée d'exclure du calcul de la MSE les premières trames de chaque séquences. Dans la suite, nous avons donc défini une nouvelle variable : n, le nombre de trames en début de séquence dont nous aimerais nous passer. La nouvelle fonction de coût devient alors :

$$MSE = \frac{1}{nSeq} \cdot \frac{1}{timesteps - n} \cdot \frac{1}{nBand} \sum_{seq=1}^{nSeq} \sum_{t=n+1}^{timesteps} \sum_{f=1}^{nBand} [M_S(seq, t, f) - M_{S, Wiener}(seq, t, f)]^2$$

i.e :

$$MSE = \frac{1}{nSeq} \cdot \frac{1}{25 - n} \cdot \frac{1}{513} \sum_{seq=1}^{nSeq} \sum_{t=n+1}^{25} \sum_{f=1}^{513} [M_S(seq, t, f) - M_{S, Wiener}(seq, t, f)]^2$$

4.4.2 Résultats avec $n = 12$

Sur la figure 4.24, les n trames que nous ne cherchons plus à prédire sont représentées en gris sur chaque séquence. Ainsi, seules les $25 - n$ dernières trames de la séquence sont utilisées pour la mesure de la fonction de coût $MSE = \frac{1}{nSeq} \cdot \frac{1}{25-n} \cdot \frac{1}{513} \sum_{seq=1}^{nSeq} \sum_{t=n+1}^{25} \sum_{f=1}^{513} [M_S(seq, t, f) - M_{S, Wiener}(seq, t, f)]^2$.

Dans un premier temps, nous avons considéré le cas où $n = 12$, qui conduit bien à la reconstruction complète de chaque séquence prédictive du masque reconstruit¹⁴.

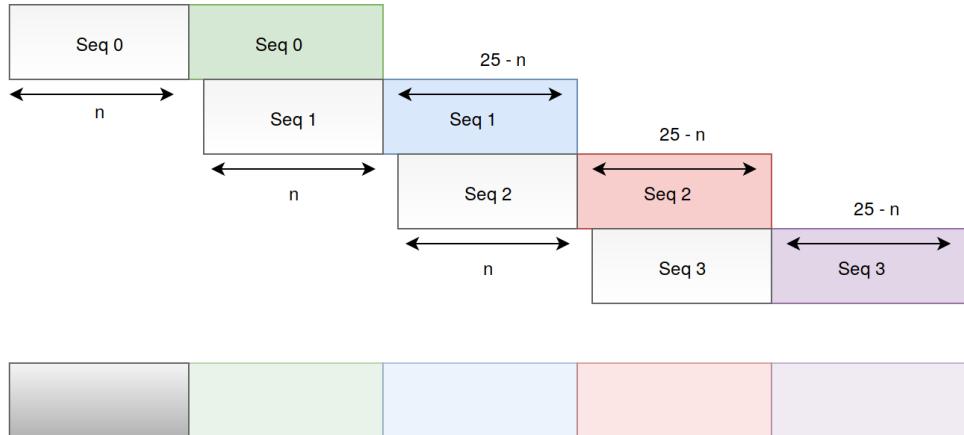


FIGURE 4.24 – Prédiction des $25 - n = 13$ dernières trames de chaque séquence

Concernant l'implémentation, il y avait plusieurs manières de faire, chacune ayant des avantages et des inconvénients. Nous avons tenté l'architecture présentée en 4.25 :

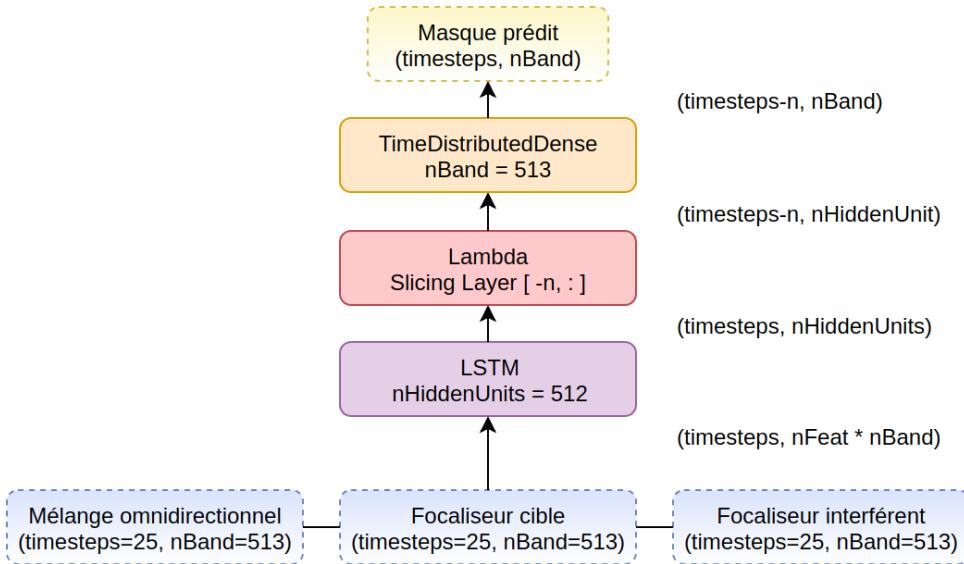


FIGURE 4.25 – Architecture du réseau estimant le masque

La couche Lambda¹⁵ définie ici permet simplement d'utiliser la fonction Lambda¹⁶ de notre choix pour définir une couche de réseau de neurones. Ici, un simple *slicing* est effectué : nous avons défini la fonction qui à une matrice de dimension ($nSeq, timesteps, nBand$) associe les $timestep - n$ dernières trames temporelles de cette même matrice.

14. À l'exception des n toutes premières trames, ce qui a été corrigé par un décalage temporel pour réaligner les masques à prédire et les séquences d'entrées.

15. <https://keras.io/layers/core/#lambda>

16. <https://docs.python.org/3/reference/expressions.html#lambda>

Les résultats obtenus sur la chaîne de traitement complètes sont étonnamment moins bon :

	Nouveau réseau, n = 12 : MSE = 0.1215	Référence : MSE = 0.1213
Source cible brute	92.6	92.8
Source réverbérée	89.7	89.7
Mélange capté	11.5	10.9
Focaliseur vers la cible	28.0	28.7
Masque de Wiener	82.7	83.9
Filtre GEVD idéal	79.4	79.5
Masque prédit	20.6	28.2
Filtre GEVD prédicts	70.4	73.7

FIGURE 4.26 – *Accuracies* du nouveau réseau avec n = 12 et du réseau de référence.

Une hypothèse probable justifiant ces résultats serait que le nouveau réseau dispose de moins de données pour l'apprentissage (13 trames au lieu de 25 pour une séquence). La partie suivante vient d'ailleurs renforcer cette hypothèse.

4.4.3 Dropout = 0.2

À la lumière de [21], il nous a semblé judicieux de réduire le *dropout* et le *recurrent dropout* dans la couche de LSTM, car la valeur précédente de 0.5 semblait un élevée, pour une première couche.

Nous avons entraîné le même réseau ne considérant pas les n premières trames, avec un *dropout* valant 0.2.

n = 12

Ici, l'*accuracy* est significativement meilleure sur les masques que pour le cas précédent, mais pas sur les filtres GEVD.

	Nouveau réseau, <i>dropout</i> = 0.2, n = 12 : MSE = 0.1223	Référence : MSE = 0.1213
Source cible brute	92.6	92.8
Source réverbérée	89.6	89.7
Mélange capté	11.3	10.9
Focaliseur vers la cible	27.7	28.7
Masque de Wiener	83.7	83.9
Filtre GEVD idéal	79.6	79.5
Masque prédit	26.1	28.2
Filtre GEVD prédicts	71.0	73.7

FIGURE 4.27 – *Accuracies* du nouveau réseau avec *dropout* = 0.2, n = 12 et du réseau de référence.

n = 3

Par acquit de conscience, nous avons également mesuré les *accuracies* dans un cas intermédiaire, n = 3, où nous utilisons plus de trames pour le calcul de la fonction de coût. La figure 4.28 illustre ce cas intermédiaire.

Les résultats en découlant sont les suivants :

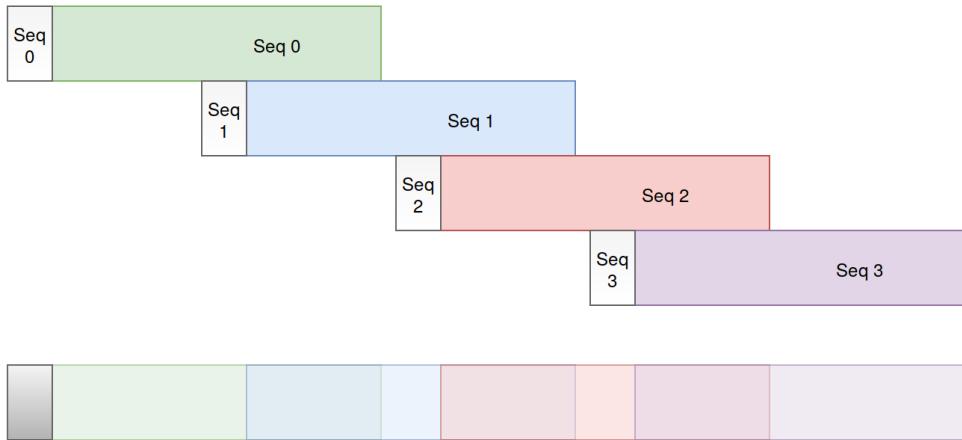


FIGURE 4.28 – Cas intermédiaire : $n = 3$

	Nouveau réseau, $dropout = 0.2$, $n = 3$: MSE = 0.1191	Référence : MSE = 0.1213
Source cible brute	92.6	92.8
Source réverbérée	89.7	89.7
Mélange capté	11.3	10.9
Focaliseur vers la cible	27.7	28.7
Masque de Wiener	83.8	83.9
Filtre GEVD idéal	79.1	79.5
Masque prédit	29.7	28.2
Filtre GEVD prédits	71.9	73.7

FIGURE 4.29 – *Accuracies* du nouveau réseau avec $dropout = 0.2$, $n = 3$ et du réseau de référence.

n = 0

Ce cas correspond donc au réseau initial de Lauréline, mais avec un *dropout* valant 0.2.

Les résultats en sont les suivants :

	<i>Dropout</i> = 0.2, $n = 0$: MSE = 0.1198	Référence (<i>Dropout</i> = 0.5) : MSE = 0.1213
Source cible brute	92.6	92.8
Source réverbérée	89.6	89.7
Mélange capté	10.8	10.9
Focaliseur vers la cible	27.5	28.7
Masque de Wiener	83.6	83.9
Filtre GEVD idéal	79.5	79.5
Masque prédit	29.3	28.2
Filtre GEVD prédits	72.3	73.7

FIGURE 4.30 – *Accuracies* du réseau de référence avec $dropout = 0.2$ et $dropout = 0.5$.

À *dropout* fixé à 0.2, les valeurs de n de 0, 3 et 12 ont donc donné des *accuracies* de 72.3, 71.9 et 71.0 : cela semble cohérent avec l'idée évoquée en fin de 4.4.2 : plus l'on utilise de trames pour une séquence donnée, meilleure semble être la reconnaissance au final.

4.5 Conclusion

Comme attendu, j'ai donc exploré diverses pistes d'améliorations du réseau que Lauréline avais dû mettre de côté : différentes méthodes de normalisation, différents canaux d'entrées, différentes méthodes de prédiction du masque en sortie. Les résultats significatifs de ces expériences sont consignés dans le tableau 4.31.

	<i>Accuracy</i>	MSE
Réseau de Lauréline (article ICASSP)	acc = 77.7	MSE indisponible
Réseau de référence (reproduction du réseau de Lauréline)	acc = 73.7	MSE = 0.1213
Canaux d'entrée normalisés de manière unique	acc = 72.6	MSE = 0.1216
Canaux d'entrée normalisés de manière unique, moyennés en fréquence	acc = 69.8	MSE = 0.1319
Canaux d'entrée de type rapports énergétiques	acc = 68.9	MSE = 0.1238
Canaux d'entrée logarithmiques	acc = 74.9	MSE = 0.1228
Nouveau réseau, n = 12	acc = 70.4	MSE = 0.1215
Nouveau réseau, dropout 0.2, n = 12	acc = 71.0	MSE = 0.1223
Nouveau réseau, dropout 0.2, n = 3	acc = 71.9	MSE = 0.1191
Réseau de référence, dropout 0.2 (n = 0)	acc = 72.3	MSE = 0.1198

FIGURE 4.31 – Résultats des tests de performance sur la partie séparation de sources

Nous retiendrons notamment l'importance de la normalisation et de l'échelles des canaux d'entrées du le réseau, ainsi que la nécessité d'avoir le plus de données d'entraînement possible pour l'apprentissage.

Avec plus de temps et une meilleure pratique de ma part, il aurait été intéressant d'explorer encore plus profondément ces pistes, notamment pour comprendre pourquoi mon modèle de référence n'a jamais atteint les performances du système de Lauréline. Cependant, ces expérimentations ont suivi une certaine logique, et la plupart des idées à vérifier ont été testées. Une exception cependant, les réseaux de neurones convolutionnels : en effet, les signaux d'entrées étant fournis au réseau sous la forme de spectrogrammes, il s'agit plus ou moins de traitement d'image. Les réseaux convolutionnels ont fait leurs preuves pour extraire des caractéristiques dans les images, et ils auraient pû être utiles pour notre tâche, pour extraire des harmoniques ou propriétés des signaux de voix.

Chapitre 5

Bilan et conclusions

Ce stage de fin d'étude m'a donc donné l'occasion de contribuer aux travaux de recherche et de développement d'Orange, sur deux aspects du traitement du signal : la localisation et la séparation de sources. Ces travaux m'ont initié à l'acoustique et m'ont fait progresser en *machine learning*.

Le VVM, algorithme de localisation développé avec MATLAB par TPS, nécessitait une implémentation en Python pour la cohérence et la maniabilité des travaux de l'équipe. C'est chose faite à l'issue de ce stage, durant lequel j'ai livré un code orienté objet reproduisant le comportant du programme original. L'algorithme de localisation implémenté a d'ailleurs évolué sur la base du code rendu, et a déjà servi à Alexandre et Srđan, pour participer au challenge LOCATA par exemple.

Mon autre mission a été la reprise et la poursuite des travaux de Lauréline sur la séparation de source. Dans le cadre de sa thèse, elle a mis au point un système de rehaussement basé sur des réseaux de neurones. Ce système, en cours de développement, nécessitait quelques améliorations, et l'ensemble des pistes d'amélioration de ses performances n'avait pas encore été exploré. À l'issue de ce stage, ces travaux sont maintenant en phase de complétion. Bien que n'ayant pas amélioré significativement les précédents résultats de Lauréline, j'ai exploré les pistes restantes et fourni une nouvelle implémentation de ce système. Cette implémentation est plus satisfaisante que la précédente, en terme de performances (gestion de la mémoire vive durant l'entraînement et l'utilisation des réseaux de neurones) et en terme de lisibilité (modularité et simplicité du code) grâce à l'utilisation de la classe *DataGenerator* de Keras. En parallèle de ma mission, Lauréline a travaillé sur un autre algorithme de localisation de sources par réseaux de neurones [16], afin de fournir les *directions d'arrivées* au système sur lequel j'ai travaillé. Après avoir interfacé ses récents travaux avec mon implémentation du système de séparation, elle va pouvoir poursuivre et achever les expérimentations nécessaires à la rédaction de son manuscrit de thèse.

En ce sens, les objectifs du stage semblent atteints pour Orange.

D'un point de vue personnel, cette expérience a été enrichissante sur de nombreux points. J'ai eu l'occasion d'appliquer mes compétences et connaissances techniques les plus fondamentales à une mission longue et cohérente. Cela m'a permis de faire un point utile sur ces acquis, et de les faire évoluer. En particulier, c'est la première fois que je dois écrire des programmes orientés objets aussi complexes. Cela m'a donné l'occasion d'expérimenter le besoin d'avoir une bonne documentation pour un programme, et également d'en écrire une aussi claire que possible, en même temps que je développe. Par ailleurs, même si j'avais, pour la localisation et pour la séparation, des illustrations concrètes de ce que mon code devait reproduire, j'ai fait face à deux contraintes différentes mais intéressante : d'une part, partir d'une « feuille blanche » pour le VVM, et d'autre part devoir comprendre et s'approprier le code de quelqu'un d'autre lors de la reprise des travaux de Lauréline.

Toutes ces expériences ont été très formatrices, et me serviront probablement dans mon futur professionnel.

Bibliographie

- [1] Mathieu Baqué. *Analyse de scène sonore multi-capteurs : Un front-end temps-réel pour la manipulation de scène*. Thèse de doctorat, Université du Maine, 2017.
- [2] Peter J. Bickel and Bo Li. Regularization in statistics. https://www.stat.berkeley.edu/~bickel/Test_BickelLi.pdf, 2006.
- [3] François Chollet et al. Keras. <https://keras.io>, 2015.
- [4] George Cybenko. Approximation by superpositions of a sigmoidal function. <https://pdfs.semanticscholar.org/05ce/b32839c26c8d2cb38d5529cf7720a68c3fab.pdf>, 1989.
- [5] Jérôme Daniel. *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*. Thèse de doctorat, 2000.
- [6] Charalampos Dimoulas, George Kalliris, Konstantinos Avdelidis, and George Papanikolaou. Improved localization of sound sources using multi-band processing of ambisonic components. In *Audio Engineering Society Convention 126*, May 2009.
- [7] Timothy Dozat. Incorporating nesterov momentum into adam. http://cs229.stanford.edu/proj2015/054_report.pdf, 2015.
- [8] Sharon Gannot, Emmanuel Vincent, Shmulik Markovich-Golan, and Alexey Ozerov. A consolidated perspective on multi-microphone speech enhancement and source separation. 2017.
- [9] G. Gravier, Jean-François Bonastre, E. Geoffrois, Sébastien Galliano, K. Mc Tait, and K. Choukri. The ester evaluation campaign for the rich transcription of french broadcast news. 01 2004.
- [10] Jahn Heymann, Lukas Drude, and Reinhold Haeb-Umbach. Neural network based spectral mask estimation for acoustic beamforming. page 196–200, 2016.
- [11] L. F. Lamel, J.-L. Gauvain, and M. Eskénazi. Bref, a large vocabulary spoken corpus for french. page 505–508, 1991.
- [12] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10 :707, 1966.
- [13] Juha Merimaa. *Analysis, Synthesis, and Perception of Spatial Sound - Binaural Localization Modeling and Multichannel Loudspeaker Reproduction*. Thèse de doctorat, 2006.
- [14] Aditya Arie Nugraha. *Deep neural networks for source separation and noise-robust speech recognition*. Thèse de doctorat, Université de Lorraine, Dec 2017.
- [15] Christopher Olah. Understanding lstm networks. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>, 2015.
- [16] Lauréline Perotin, Romain Serizel, Emmanuel Vincent, and Alexandre Guérin. Crnn-based multiple doa estimation using ambisonics acoustic intensity features. Jul 2018.
- [17] Lauréline Perotin, Romain Serizel, Emmanuel Vincent, and Alexandre Guérin. Multichannel speech separation with recurrent neural networks from high-order ambisonics recordings. page 1–5, Apr 2018.
- [18] Tim Peters. Pep 20 – the zen of python. <https://www.python.org/dev/peps/pep-0020/>, 2004.
- [19] Lutz Prechelt. Early stopping - but when? http://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf.
- [20] Daniel Shin. Overfitting & regularization. <http://dshinccd.github.io/blog/regularization/>, 2015.
- [21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. 2014.
- [22] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. <http://www.cs.toronto.edu/~fritz/absps/momentum.pdf>.
- [23] Andy Thomas. Keras lstm tutorial : How to easily build a powerful deep learning language model. <http://adventuresinmachinelearning.com/keras-lstm-tutorial/>, 2018.
- [24] Ziteng Wang, Emmanuel Vincent, Romain Serizel, and Yonghong Yan. Rank-1 constrained multichannel wiener filter for speech recognition in noisy environments. page 37–51, 2018.