Traitement de la phase des signaux audio dans les réseaux de neurones profonds

Félix Mathieu

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT046

Thèse de doctorat

Traitement de la phase des signaux audio dans les réseaux de neurones profonds

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 28 novembre 2023, par

FÉLIX MATHIEU

Composition du Jury :

Sylvain MARCHAND Professeur, Université de La Rochelle	Président/Examinateur
Lourent Oudre Professeur, ENS Paris Saclay	Rapporteur
Matthieu Kowalski Professeur associé, Université Paris-Sud	Rapporteur
Alice Cohen Hadria Maîtresse de conférence, Sorbonne Université	Examinatrice
Geoffroy Peeters Professeur, Télécom Paris	Directeur de thèse
Gaël Richard Professeur, Télécom Paris	Co-directeur de thèse
Thomas Courtat Thales SIX	Invité

Télécom Paris

Thèse de doctorat

Traitement de la phase des signaux audio dans les réseaux de neurones profonds

---

Auteur :
Félix Mathieu

Directeurs de thèse :
Geoffroy Peeters,
Gaël Richard

Thèse de doctorat de l'Institut Polytechnique de Paris préparée à

ADASP group (Audio Data Analysis and Signal Processing)
Signal, Statistique et Apprentissage

TÉLÉCOM PARIS

Résumé

Télécom Paris
Signal, Statistique et Apprentissage

Docteur

Traitement de la phase des signaux audio dans les réseaux de neurones profonds

Les algorithmes de séparation de sources audio reposent pour une grande part sur la Transformée de Fourier à Court Terme (TFCT). Cependant, la TFCT est munie d'une composante de phase variant rapidement et à l'allure peu structurée, difficile à modéliser pour des algorithmes de traitement du signal classiques. Ces dernières années, les approches reposant sur les réseaux de neurones profonds ont permis de traiter différentes variantes de la séparation de sources avec succès. Cependant, la reconstruction de la phase des signaux reste tout de même une tâche difficile pour ce type d'algorithme. L'objectif de cette thèse est de construire et d'évaluer des méthodes permettant de modéliser la phase des signaux dans les réseaux de neurones profonds pour la séparation de sources audio.

Dans un premier temps, nous examinons les réseaux de neurones à valeurs complexes qui devraient permettre d'analyser et de synthétiser la phase de la TFCT. Nous montrons que, dans l'état de l'art actuel, d'une part l'utilisation naïve des réseaux de neurones complexes ne permettent de créer des représentations internes convenables de la phase dans leurs états intermédiaires ; d'autre part, que forcer une structure complexe entraîne une baisse de performance importante sur une tâche de séparation de plusieurs voix parlées.

Dans un second chapitre, nous changeons de paradigme et modélisons les signaux à travers des représentations réelles calculées de manière similaire à la TFCT. Ces représentations peuvent être calculées en convoluant un signal audio directement avec un banc de filtres réels comme cela est proposé dans le modèle de séparation de sources TasNet. Nous établissons une liste de propriétés désirables pour ces filtres (invariances aux décalages temporels et de phase) pour que les représentations induites soient plus facilement interprétables et augmentent les performances du modèles. Nous proposons de paramétrer ces filtres en garantissant les propriétés définies tout en essayant de conserver de la flexibilité lors de l'apprentissage. Nous vérifions nos hypothèses sur une tâche de séparation de sources voix-voix identique à celle étudiée dans le chapitre sur les réseaux de neurones à valeurs complexes, mais aussi sur une tâche de rehaussement de la parole. Expérimentalement, nous montrons que ces bancs de filtres permettent d'obtenir des performances égales ou supérieures aux paramétrisations actuelles de

l'état de l'art tout en offrant une interprétation accrue de l'espace de représentation du signal dans la première couche du modèle et permettant une augmentation de la taille des fenêtres d'analyse pour ce type d'architecture.
Ces deux premières études modélisent la phase du signal à partir de projections dans des espaces temps-fréquences ou assimilés. Dans le dernier chapitre, nous travaillons directement sur la forme d'onde temporelle à travers une variante du modèle Wav-UNet qui consiste en une succession de convolutions 1D appliquées tour à tour sur des versions sous-échantillonnées du signal. Notre apport consiste à remplacer les convolutions du modèle par des convolutions séparables et de remplacer les parties *depthwise* avec des filtres de Gabor. Ce faisant, on garantit à tous les étages du modèle des représentations bien localisées en temps et en fréquences. Cette paramétrisation permet également de diminuer le coût de calcul et le nombre de paramètres du modèle. Nous montrons que les fréquences centrales des filtres appris convergent vers des distributions logarithmiques en fréquences à tous niveaux du modèle. Les expériences sont effectuées sur une tâche de réhaussement de la parole identique à celle présentée dans le chapitre sur les bancs de filtres.
Ainsi, dans cette thèse, nous proposons plusieurs méthodes pour modéliser et synthétiser au mieux la phase dans les réseaux de neurones : soit par l'utilisation de structures adéquates (réseaux de neurones complexes, représentations invariantes à la phase) ; soit par l'utilisation de modèles traitant directement le signal dans le domaine temporel.

Remerciements

La réalisation de cette thèse a été rendue possible grâce à la multitude d'échanges et de rencontres de ces dernières années, dont la contribution à la réussite de ce projet de recherche est inestimable.

Je souhaite tout d'abord exprimer ma reconnaissance particulière envers mes directeurs de thèse, Geoffroy Peeters et Gaël Richard, dont l'expertise et le soutien ont été des piliers essentiels à la concrétisation de ce projet intellectuel. Leur accompagnement, surtout dans les moments plus délicats de cette démarche, a été inestimable. Mes remerciements s'étendent également aux membres de l'équipe ADASP, qui ont constitué un écosystème collaboratif précieux au cours de ces trois années.

Un hommage particulier revient à Thomas, mon superviseur industriel chez Thales, dont la confiance a été un moteur crucial de mon parcours de doctorant. Travailler sous ta supervision a été une expérience exceptionnelle, enrichissant autant mes compétences techniques que méthodologiques. Ta capacité à transmettre tes connaissances avec patience et clarté a joué un rôle central dans mon apprentissage. Je suis aussi reconnaissant pour tous les conseils éclairés que tu as pu me faire lors des situations les plus complexes de mon parcours. C'est avec une profonde gratitude que je termine cette étape académique, honoré d'avoir pu évoluer aux côtés d'une personne aussi inspirante que toi.

Je saisis également cette occasion pour exprimer ma reconnaissance envers mes collègues de Thales, Léo, Shaheen, Ludovic, François, Bilal, et bien d'autres, qui ont partagé abondamment leurs connaissances et leur bienveillance au fil de ces trois années.

Mes remerciements s'étendent à mes amis, Victor, Armand, Brieuc, Antonin, Loup, Lucas, Théophile, Maxime, qui ont été des sources de soutien indéfectible et ont constamment nourri ma passion à travers le partage incessant de leurs centres d'intérêt variés. Enfin, je tiens à remercier ma famille et plus particulièrement mes parents pour leur soutien émotionnel et moral. Je voudrais également profiter de cette page pour remercier mon grand-père d'avoir été un soutien essentiel durant l'élaboration de cette thèse. Sa présence a été un facteur déterminant dans la réalisation de ce projet, et je suis honoré de pouvoir perpétuer cette tradition de chercheur au sein de notre famille.

Table des matières

Table des figures

Liste des tableaux

Liste des Abréviations

ASR	Automatic Speech Recognition
BN	Batch Normalisation
CLU	Compressor Linear Unit
CNN	Convolutional Neural Network
CTN	Conv-TasNet
DCCRN	Deep Complex Conolutional Reccurent Neural Networks
DNN	Deep Neural Network
DNS	Deep Noise Suppression Challenge
FLOPS	FLoating-point Operations Per Second
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory
MLP	Multi-Layer Peceptron
MPGTF	Multi-Phase Gammatone FilterBank
PESQ	Perceptual Evaluation of Speech Quality
RNN	Recurrent Neural Netowrk
RNVC	Réseaux de Neurones à Valeurs Complexes
SDR	Source to Distorsion Ratio
Si-SNR	Scale-invariant Source to Noise Ratio
STOI	Short-Time Objectiv Intelligibility
TCN	Temporal Convolutional Network
TF	Transformée de Fourier
TFD	Transformée de Fourier Discrète
TFCT	Transformée de Fourier Court Terme

Liste des Symboles

Transformations et opérations

$\mathcal{F}$	Transformée de Fourier discrète
$\mathcal{H}$	Transformée de Hilbert
$\mathcal{H}_\psi$	Transformée de Hilbert étendue
TFCT	Transformée de Fourier Court-Terme
$\circledast$	Produit de convolution
$\odot$	Produit d'Hadamard

Signaux

$s(t)$	Signal dans le domaine temporel
$S(f)$	Signal dans le domaine fréquentiel
$\hat{s}(t)/\hat{S}(f)$	Signal estimé
$\tilde{s}(t)$	Signal analytique
ν	Fréquence normalisée
ϕ	Phase
ψ	Décalage de phase
RG	Retard de groupe
FI	Fréquence instantanée (pour la TFCT)
$\dot{\phi}$	Fréquence instantanée (pour un signal analytique)

Modèles

f_e	Encodeur
f_d	Décodeur
f_s	Séparateur

Chapitre 1

Introduction

Comment peut-on séparer efficacement les différentes sources sonores dans un enregistrement audio donné, afin d'en extraire des pistes individuelles pour des sources sonores d'intérêt ? Pour capturer un signal sonore particulier, des composantes parasites, structurées ou non, peuvent interférer et diminuer les capacités de compréhension et de perception de celui-ci. Dans le cas où le signal ciblé correspond à un signal de voix, ces détériorations peuvent rendre incompréhensible le message transmis par le locuteur enregistré. On peut également vouloir séparer dans plusieurs canaux dédiés les différentes informations d'un signal audio composé de plusieurs éléments distincts, par exemple lorsque deux personnes parlent en même temps. Ces problématiques apparaissent typiquement lors de la transmission de discussion via des applications de discussion en ligne. Comment mesurer les performances d'un modèle de séparation de sources ? Il peut être question de la qualité de reconstruction des pistes individuelles obtenues à partir du signal mélangé, on parlera aussi d'intelligibilité dans le cas des signaux de parole. Mais, il peut aussi être question de la résilience des modèles face à des interférences et des perturbations présentes dans le mélange, qui peuvent rendre la séparation plus difficile (saturation, réverbération, perte de paquets, interférence). On peut également utiliser comme métrique de comparaison des algorithmes la réduction du coût computationnelle ou l'interprétabilité de leurs traitements.

1.1 Motivations et objectifs

Le rehaussement de la parole est la tâche qui consiste à améliorer la qualité et l'intelligibilité de la voix dans un signal bruité. La séparation de sources consiste quant à elle à estimer différents éléments d'un mélange de signaux dans des canaux dédiés. On peut voir la tache de rehaussement de la parole comme un cas particulier de la séparation de sources. Historiquement, les algorithmes utilisés pour résoudre ces deux tâches sont de nature différente. En effet, pour le rehaussement de la parole, on peut par exemple faire appel à des a priori sur la constitution d'un signal de voix pour sélectionner et rehausser les composantes de celui-ci. Dans le cas de la séparation de sources, ce genre d'a priori ne peut pas toujours être utilisé, par exemple dans le cas de la séparation de deux voix distinctes.

L'émergence des réseaux de neurones profonds a permis de traiter ces deux problématiques dans un même formalisme. L'utilisation de bases de données massives permet en effet de laisser le modèle construire des a priori sur les données sans intervention experte. L'apprentissage profond a permis d'améliorer considérablement les performances de ces deux tâches, mais au prix du coût computationnel et de l'interprétabilité des algorithmes. L'objectif de cette thèse est justement d'améliorer l'interprétabilité des architectures de réseaux de profonds tout en essayant de limiter le coût computationnel de ceux-ci.
On s'interessera tout particulièrement à la modélisation de la phase des signaux dans ces algorithmes. Les composantes de phases des signaux restent aujourd'hui difficiles à modéliser même pour des réseaux de neurones profonds. Pour cela, on approfondira et étendra plusieurs stratégies pour modéliser la phase des signaux dans les réseaux de neurones profonds :

— L'utilisation de structures à valeurs complexes,
— L'utilisation de représentations invariantes à la phase,
— L'utilisation de modèles permettant de s'abstraire de la composante de phase.

Tous ces points seront élaborés en s'assurant de conserver ou de diminuer le coût computationel d'algorithme existant et d'augmenter l'interprétabilité des décisions d'algorithmes existants.

1.2 Structure de la thèse

Ce document est divisé en deux parties. Une première partie est dédiée à la présentation de l'état de l'art et des outils de traitement du signal et d'apprentissage machine nécessaires à la bonne compréhension des travaux effectués dans le cadre de cette thèse. La seconde partie est consacrée aux différentes contributions et travaux menés. Nous détaillons ci-dessous l'organisation des chapitres de ces deux parties.

Partie 1 : Cadre théorique et état de l'art

— **Chapitre 2 : Traitement du signal** Ce chapitre permet d'introduire des notions de traitement du signal fondamentales pour comprendre les outils utilisés et les enjeux de modélisation étudiés dans la deuxième partie.
— **Chapitre 3 : Apprentissage profond** Ce chapitre présente les architectures et les stratégies d'optimisation de bases des réseaux de neurones profonds. De plus, nous présenterons également en détails les particularités des réseaux de neurones à valeurs complexes.
— **Chapitre 4 : Séparation de sources monaurale et rehaussement de la parole** Dans ce chapitre, nous présentons d'abord les tâches de séparation de sources et de rehaussement de la parole, puis nous présentons les principaux algorithmes d'apprentissage profond permettant de les résoudre. Nous en profiterons également pour présenter les fonctions d'évaluation et les bases de données utilisées dans les différents chapitres de la seconde partie.

Partie 2 : Contributions

- **Chapitre 5 : Analyse des réseaux de neurones à valeurs complexes** Dans ce chapitre, on étudie les capacités des réseaux de neurones à valeurs complexes pour modéliser la phase des signaux audio. On discute en particulier de l'intérêt d'utiliser ce type d'architectures sur des algorithmes basés sur la transformée de Fourier court terme et sur des stratégies bout-en-bout.
- **Chapitre 6 : Apprentissage de banc de filtres experts** Dans ce chapitre, on s'intéresse à des modélisations des signaux analogues à celles de la TFCT permettant de maximiser les performances des modèles et l'interprétabilité des représentations apprises. On établit un ensemble de propriétés apparaissant naturellement dans l'apprentissage de modèles bout-en-bout et on cherche à contraindre des modèles à les obtenir naturellement par la construction de banc de filtres paramétrés.
- **Chapitre 7 : Réseaux de neurones à convolutions paramétrées** Dans ce chapitre, on présente une paramétrisation d'un réseau de neurones convolutionnel afin que l'ensemble de ses représentations soit interprétable.
- **Conclusion** Enfin, le dernier chapitre permet de résumer l'ensemble des travaux présentés dans cette thèse et de proposer de nouvelles perspectives afin de prolonger ces travaux.

1.3 Contributions et publications

L'objectif de cette thèse est de modéliser la phase des signaux dans des réseaux de neurones profonds ; soit par l'utilisation de structures internes au modèle propice à cette modélisation (chapitre 5 et chapitre 6), soit par l'utilisation de modèles bout-en-bout dans le domaine temporel permettant de s'affranchir de la phase (chapitre 7). Les chapitres 6 et 7 ont fait l'objet de publications détaillées ici :

- **Chapitre 6 : Apprentissage de banc de filtres experts** Ce chapitre a fait l'objet de deux publications :
  - Une publication dans la conférence internationale ICASSP de 2022 avec l'article *Phase-Shifted Bedrosian Filterbank : An interpretable audio front-end for the time-domain audio source separation*. Dans cet article, on détaille la paramétrisation d'un banc de filtres permettant de garantir l'invariance à des décalages de phases et des décalages temporels lors de l'application de celui-ci sur un signal audio d'entrée. De plus, on présente également une extension de la transformée de Hilbert permettant de sur-paramétrer la phase de sous-ensembles de filtres. On montre que cette surparamétrisation permet d'améliorer les performances de modèle de type TasNet tout en réduisant le nombre de paramètres.
  - Une pubication dans la conférence française GRETSI de 2022 avec l'article *Apprentissage de banc de filtres pour la séparation aveugle de sources sonores.*

Dans cet article, on donne des détails théoriques sur l'assurance d'avoir certaines propriétés d'invariances dans les bancs de filtres construits. Cet article constitue une sorte d'annexe à l'article précédent.

— **Chapitre 7 : Réseaux de neurones à convolutionnels paramétrées** Ce chapitre a fait l'objet d'une publication :
  — Une publication dans la conférence internationale ICASSP 2023 avec l'article *Learning interpretable filters in Wav-UNet for speech enhancement*. Dans cet article, on étend la représentation du signal en banc de filtres des articles précédents à toutes les couches d'un réseau de neurones. En effet, on présente une paramétrisation des convolutions d'un modèle de telle sorte à obtenir des représentations passe-bandes dans le modèle. On garantit une certaine régularité dans la distribution fréquentielle des différentes représentations internes du modèle. Ainsi, on permet potentiellement d'utiliser des transformations expertes de traitement du signal à tous les niveaux de cette architecture.

Première partie

Cadre théorique et état de l'art

Chapitre 2

Traitement du signal

Dans ce chapitre, nous rappelons la définition d'objets couramment utilisés pour modéliser un signal audio, l'analyser et le valoriser. Par nature, un signal audio est un phénomène continu. Pour le traiter d'un point de vue informatique, il faut le transformer en une représentation discrète. Nous présentons donc ici la transformée de Fourier discrète ainsi que des propriétés et des extensions de celle-ci. Nous en profiterons pour introduire les notations qui seront observées dans la suite de cette thèse.

2.1 Transformée de Fourier discrète

Intuitivement, un signal est la mesure d'une quantité qui varie avec le temps. On considère deux représentations des signaux : les signaux analogiques ou continus qui sont en général des objets physiques et les signaux numériques qui permettent le traitement et le stockage par l'informatique.

2.1.1 Signal numérique

Un *signal numérique* x est une séquence discrète que l'on note :

$$x = \{x[n]\}_{n\in\mathbb{Z}}. \tag{2.1}$$

On peut construire un signal discret x à partir d'un signal continu x_c en prenant :

$$x[n] = x_c(nT_e), \tag{2.2}$$

on dit que x est une version échantillonnée de x_c et le paramètre T_e est appelé période d'échantillonnage. Son inverse $f_s = 1/T_e$ est appelé la *fréquence d'échantillonnage*.
Un signal de voix échantillonné à 44.1 kHz est présentée dans la figure 2.1. Hormis l'information d'amplitude générale du signal, le reste de l'information est difficilement visualisable dans ce domaine. On peut cependant observer dans la sous-figure de droite que des motifs répétitifs apparaissent sur des échelles de temps plus courtes.

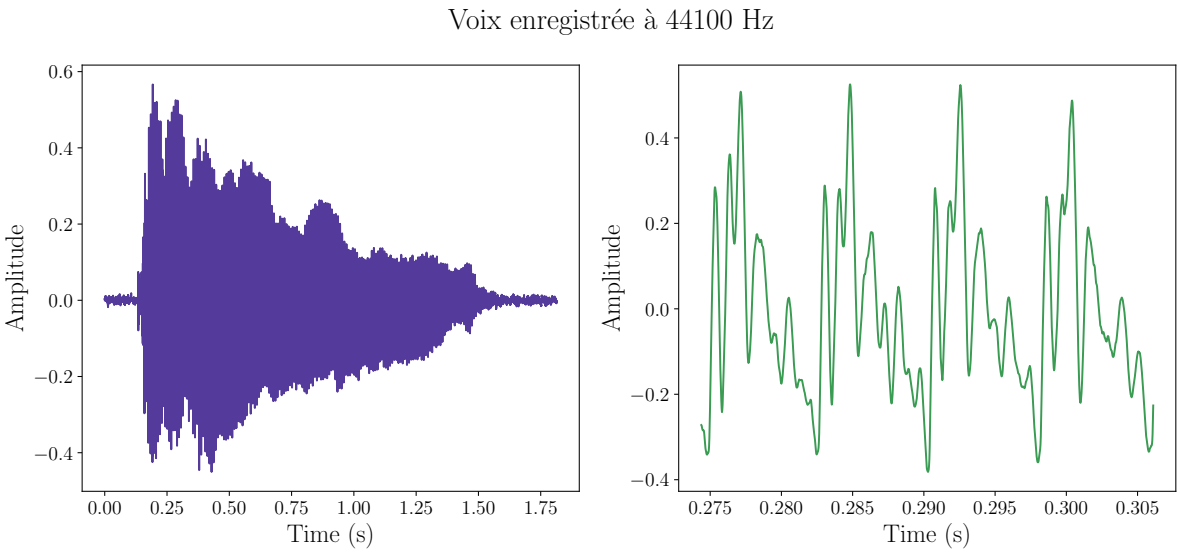


FIGURE 2.1 – Visualisation d'une voix humaine. La figure de gauche correspond au signal numérique d'une voix prononçant le phonème /a/ sur une seconde et demi. La figure de droite correspond à un zoom de cette même voix, sur une échelle de temps de l'ordre de la dizaine de millisecondes ; on peut y observer la répétition d'un motif.

2.1.2 Transformée de Fourier pour les signaux discrets

La transformée de Fourier discrète (TFD) est un outil permettant de représenter un signal discret fini dans une base selon un autre point de vue que le point de vue temporel et de mettre en exergue des phénomènes comme la périodicité des motifs observés dans la figure 2.1.
Numériquement, on ne manipule jamais de signaux continus infinis, mais une version échantillonnée et finie $x[n]$ avec $n \in [\![0, N-1]\!]$, la version échantillonnée d'un signal continu x_c. Dans ce contexte, on définit la transformée de Fourier discrète par :

$$\mathcal{F}(x)[k] \triangleq X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi j \frac{nk}{N}}. \tag{2.3}$$

La TFD est inversible, et l'on peut reconstruire la séquence x à partir des coefficients $X[k]$ en prenant :

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{2\pi j \frac{kn}{N}}. \tag{2.4}$$

Cette définition de la transformée de Fourier pour les signaux finis discrets est un cas particulier de la transformée de Fourier à temps discret, dont la séquence x de taille N est assimilée à un signal $\tilde{x}$ avec $\tilde{x}[t] = 0$ pour $t \in \mathbb{Z} \backslash [0, N-1]$ [1].

En notant $w_N = e^{2\pi j/N}$, notons que la TFD peut être vue comme une application linéaire de $\mathbb{C}^N$ dans $\mathbb{C}^N$ définie par la matrice :

$$\begin{pmatrix} w_N^{0\times 0} & \dots & w_N^{0\times N-1} \\ \vdots & w_N^{k\times n} & \vdots \\ w_N^{N-1\times 0} & \dots & w_N^{N-1\times N-1} \end{pmatrix} \quad k, n \in [\![0, N-1]\!]. \tag{2.5}$$

Cette matrice est notamment unitaire et donc inversible. Chaque ligne de celle-ci correspondent à une sinusoïde de fréquence $\frac{k}{N}$ échantillonnée, une fréquence de 1.

Convolution et transformée de Fourier

L'opération de convolution entre deux signaux discrets x et r de $\mathbb{C}^N$ est par définition :

$$(x \circledast r)[n] = \sum_{m=-\infty}^{+\infty} x[n-m]r[m]. \tag{2.6}$$

La transformée de Fourier permet de modifier les convolutions du domaine temporel en produits du domaine fréquentiel, et, réciproquement, les produits du domaine temporel en convolutions du domaine fréquentiel.

$$\mathcal{F}(x \circledast r)[k] = \mathcal{F}[k].\mathcal{F}[r], \tag{2.7}$$

2.1.3 Limitations de la TFD

La TFD permet d'obtenir une décomposition en fréquence d'une séquence x, ce qui donne une vision globale du signal (chaque échantillon de la TFD correspond à l'amplitude et à la phase d'une sinusoïde de même longueur que le signal analysé) tandis que la vision temporelle offre une vision locale du signal. Cependant, plusieurs paramètres limitent les fréquences observables dans la TFD.

Fréquences observables sur un signal échantillonné

La fréquence d'échantillonnage donne une limite sur les fréquences maximales observables. Le *théorème de Nyquist-Shannon* [2] stipule qu'un signal analogique réel peut exactement être interpolé à partir du signal échantillonné $x[n] = x_c(\frac{n}{f_s})$ si le support de la TF de x est inclu dans un intervalle $[-f_s/2, f_s/2]$. On dit de x qu'il est à bande limitée. Si la version continue x_c du signal x présente une composante en fréquence $f_+ > \frac{f_s}{2}$, cette dernière ne pourra pas être analysée correctement dans la TFD de x. Cette composante fréquentielle modifiera cependant x et sa TFD sur d'autres fréquences. Ce phénomène s'appelle l'*aliasing*.

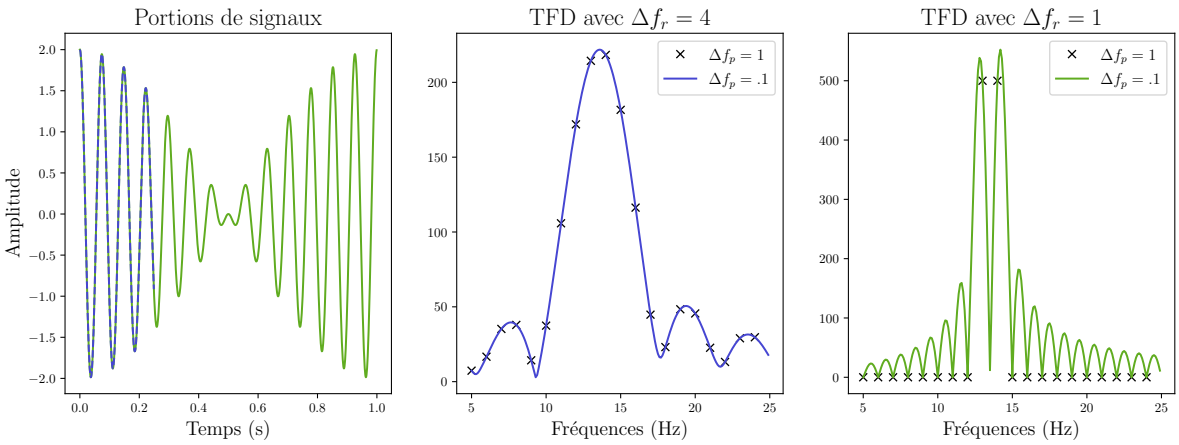


FIGURE 2.2 – (a) Signal fini composé de deux sinusoïdes pures de fréquence 13 et 14 Hz sur une durée d'observation d'une seconde échantillonné à 1000Hz. (b) TFD du signal pour une durée d'observation .25 seconde. En noir, la TFD avec une précision de $\Delta f_p = 1$, en bleu la TFD avec une précision de $\Delta f_p = .1$. Les deux fréquences ne sont pas distinguables et on a $\Delta f_r = 4$.(c) TFD du signal pour une durée d'observation d'une seconde. En noir, la TFD avec une précision de $\Delta f_p = 1$, en vert la TFD avec une précision de $\Delta f_p = .1$. Les deux fréquences y sont cette fois-ci distinguables et on a $\Delta f_r = 1$.

Précision et résolution

La résolution fréquentielle de la TFD d'un signal est définie à partir de la durée d'observation N du signal x et de la fréquence d'échantillonnage du signal f_s par la formule :

$$\Delta f_r = \frac{f_s}{N}. \tag{2.8}$$

Cette valeur correspond à l'écart minimal en fréquence à avoir afin que deux sinusoïdes de fréquences différentes soient distinguables dans la TFD de leur somme.
La précision fréquentielle de la TFD correspond à l'échantillonnage de la TFD. Pour un signal de taille M échantillonné à la fréquence f_s on a :

$$\Delta f_p = \frac{f_s}{M}. \tag{2.9}$$

On peut cependant modifier la précision de la TFD en ajoutant des zéros au signal x afin d'obtenir un signal $\tilde{x}$ de taille M'. Ce faisant, on calcule la TFD avec un plus grand nombre de sinusoïdes ayant des fréquences plus proches. Augmenter la précision permet d'affiner la représentation fréquentielle du signal. Cependant, cette augmentation ne permet pas d'améliorer la résolution fréquentielle, on l'utilise principalement pour améliorer la détection des pics fréquentiels du signal.
La figure 2.2 présente un signal composé de deux sinusoïdes et la transformée de Fourier discrète de celui-ci selon deux niveaux de résolution. L'une ne permettant pas la distinction des deux fréquences, l'autre la permettant.

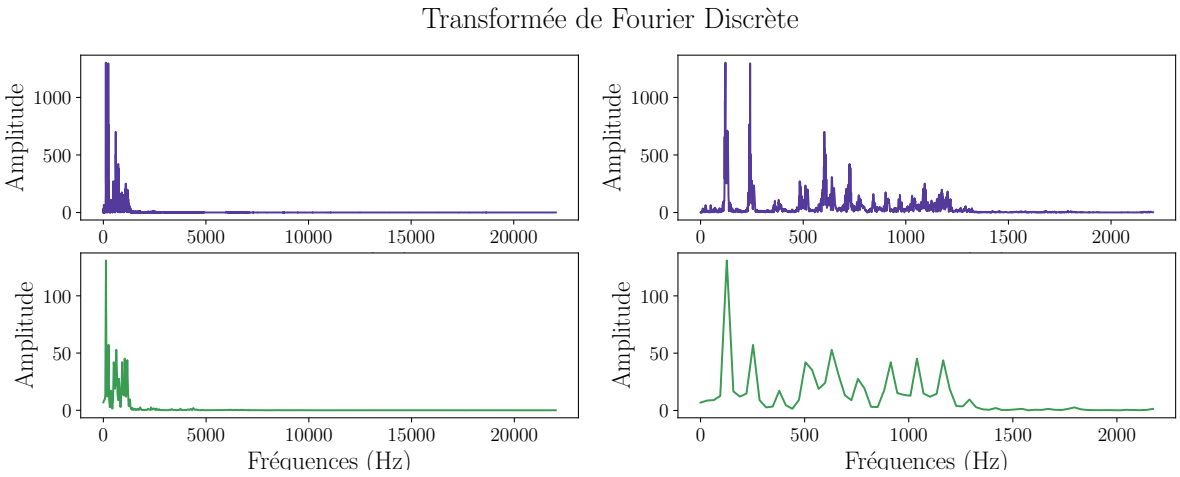


FIGURE 2.3 – Transformée de Fourier discrète du signal de la figure 2.1. En bleu la TFD du signal entier et en vert la TFD du signal tronqué. À gauche, la TFD complète du signal, à droite la TFD sur la plage de fréquences $[0, 2200]$.

2.2 Transformée de Fourier Court Terme

La TFD est adaptée à décrire des signaux stationnaires dans le temps, mais capture difficilement des variations de régime dans le temps [3]. Dans cette partie, on va décrire la transformée de Fourier court terme (TFCT) permettant de projeter un signal discret dans les domaines temporels et fréquentiels conjointement.

2.2.1 Définitions

Fenêtre d'analyse

On appelle *fenêtre d'analyse* une fonction w_a réelle à valeurs dans $\mathbb{R}$ dont ses valeurs sont nulles en dehors d'un intervalle I appelé support de la fenêtre :

$$w_a(t) = \begin{cases} w(t) & \text{si } t \in I, \\ 0 & \text{sinon,} \end{cases} \tag{2.10}$$

avec w une fonction continue et strictement positive sur I. L'intervalle I est par convention de la forme $[0, L]$. On dit de W que c'est une fenêtre de longueur L.

TFCT

Pour un signal discret $x \in \mathbb{R}^T$, une fenêtre w_a de longueur L et un pas d'avancement s, on définit la TFCT de x par :

$$\text{TFCT}(x)(n,k) = \sum_{m=0}^{L-1} x(sn+m).w_a(m)e^{-2\pi j \frac{km}{L}}, \text{ pour } k \in [\![0, L-1]\!] \text{ et } ns \leq T - L \tag{2.11}$$

La TFCT correspond donc à une succession de transformées de Fourier discrètes sur les segments $x(ns : ns + L)$. La TFCT est une représentation complexe, on peut la considérer comme la somme de ses parties réelles et imaginaires :

$$\mathrm{TFCT}(x)(n,k) = a_x(n,k) + jb_x(n,k), \tag{2.12}$$

où $a_x(n,k)$ et $b_x(n,k)$ correspondent respectivement aux produits scalaires du signal x par le cosinus et le sinus de fréquence k/L. Mais on peut également considérer la TFCT par son amplitude et sa phase ϕ :

$$\mathrm{TFCT}(x)(n,k) = A_x(n,k)e^{-j\phi_x(n,k)}. \tag{2.13}$$

L'amplitude $A_x(n,k)$ est appelée spectrogramme.
La figure 2.4 montre la TFCT d'un signal de parole factorisé en une partie d'amplitude et une partie de phase.

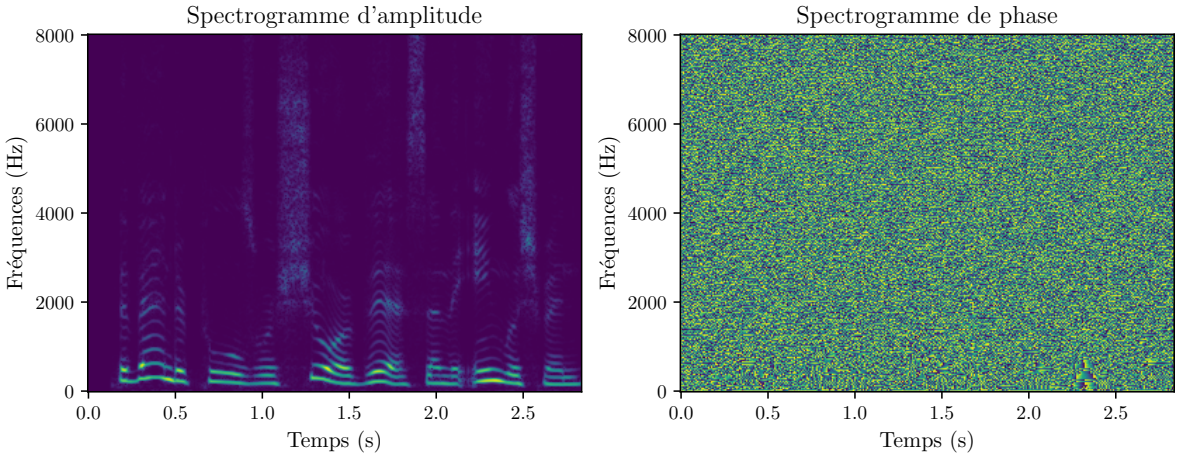


FIGURE 2.4 – Spectrogramme d'amplitude et spectrogramme de phase d'un signal de parole. Le spectrogramme de phase semble aléatoire et ne permet pas de distinguer précisément la présence de la voix.

Inversibilité

La TFCT est une transformation inversible. On peut retrouver le signal d'origine x en prenant la TFD inverse sur les différentes trames $n \in [\![0, T/s]\!]$. Une fois les portions de signaux reconstruites, on utilise une procédure d'addition-recouvrement (*overlapp-and-add* en anglais) pour prendre en compte les superpositions induites par le pas d'avancement.

2.2.2 Paramètres d'analyses

Choix de la fenêtre d'analyse

Lorsque l'on calcule la TFD sur les différentes portions du signal étudié, le choix de la fonction de fenêtrage w_a est crucial. En effet, quelle que soit la fenêtre w_a utilisée, des effets de bords plus ou moins importants apparaissent dans la projection résultante. Ces effets de bord s'expliquent très bien dans le domaine fréquentiel. En effet, la multiplication d'un signal par une fonction de fenêtrage finie est équivalent dans le domaine fréquentiel à convoluer la réponse fréquentielle du signal par la transformée de Fourier de la fenêtre w_a. Selon la forme de $\mathcal{F}(w_a)$, cette convolution peut plus ou moins distordre la réponse fréquentielle du signal et entraîne une modification de la résolution fréquentielle. Par exemple, dans le cas où le signal étudié correspondrait à une sinusoïde pure et où w_a correspondrait à la fonction porte suivante :

$$\Pi(t) = \begin{cases} 1 & \text{si } t \in [0, L], \\ 0 & \text{sinon,} \end{cases} \tag{2.14}$$

l'amplitude de la transformée de Fourier de cette sinusoïde sur la fenêtre w_a apparaîtra comme la valeur absolue d'un sinus cardinal.

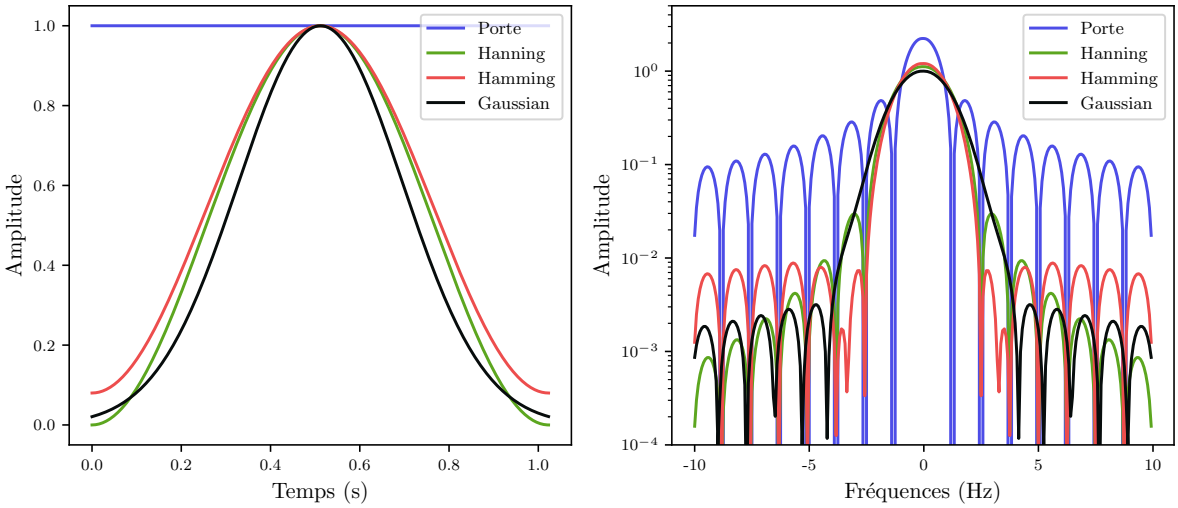


FIGURE 2.5 – Différentes fenêtres d'analyse w_a associées à leurs transformées de Fourier respectives.

Les lobes présents dans la TF de la fenêtre d'analyse se décomposent en deux catégories :

- Le lobe principal qui correspond au lobe autour de la fréquence 0,
- Les lobes secondaires qui correspondent à tous les autres lobes.

La largeur du lobe principal donne une information sur la résolution de la TF, tandis que les lobes secondaires, plus petits que le lobe principal, causent des interférences dans les résultats de l'analyse.

Différentes fenêtres sont utilisés en analyse spectrale en analyse spectrale, chacune ayant des caractéristiques différentes (largeur du lobe principal, amplitude des lobes

secondaires, etc...). Les fenêtres les plus couramment utilisées sont la fenêtre Gaussienne, de Hamming ou de Hann. Pour t dans $[0, L]$, on a :

$$w_{\text{Gauss}}(t) = e^{-\frac{1}{2}\left(\frac{t-L/2}{\sigma L/2}\right)^2} \quad \sigma \leq 0.5, \tag{2.15}$$

$$w_{\text{Hann}}(t) = 0.5 - 0.5\cos(2\pi\frac{t}{L}), \tag{2.16}$$

$$w_{\text{Hamming}}(t) = 0.54 - 0.46\cos(2\pi\frac{t}{L}). \tag{2.17}$$

Le choix de l'une ou l'autre de ces fenêtres est le résultat d'un compromis entre la réduction de l'amplitude des lobes secondaires et de la distorsion de la réponse fréquentielle principale induite par la largeur du lobe principale, ce qui est une façon d'illustrer les relations entre précision et résolution [3]. La figure 2.5 présente des fonctions de fenêtrages standards ainsi que leurs transformées de Fourier respectives.

Taille de la fenêtre

La taille de la fenêtre L joue un rôle important dans la résolution du domaine fréquentiel. Il est aussi à noter qu'il n'existe pas de taille de fenêtre optimale pour la TFCT. Cette discussion rejoint la problématique de la résolution en fréquence présentée dans le paragraphe 2.1.3.

- Pour des fenêtres d'analyse trop petites, la discrimination des fréquences est difficile.
- Pour des fenêtres d'analyse trop grandes, la localisation temporelle des évènements dans le signal est perdue (dans le cas de la voix, on peut par exemple, avoir la présence de plusieurs phonèmes dans la même fenêtre d'analyse). Même si cette information est présente dans la phase de la transformée de Fourier, elle reste difficile à analyser.

Par exemple, dans le cas d'un signal de parole, on aimerait associer aux sons plausifs (sons très localisés en temps) des fenêtres de petites tailles tandis qu'on aimerait associer aux sons fricatifs et voisés (signaux quasi stationnaires) des fenêtres assez longues afin d'obtenir une représentation fréquentielle aussi précise que possible.
Un signal de voix étant par nature une succession de ces différents types de sons, il faudra analyser le signal sur des fenêtres suffisamment petites pour différencier les sons successifs, mais suffisamment grandes pour obtenir une représentation fréquentielle claire.
Le choix de la taille de la fenêtre est donc un élément crucial qui résulte d'un compromis lorsque l'on fait une analyse temps-fréquence avec la TFCT. Il dépend à la fois des signaux étudiés, mais aussi de ce que l'on cherche à analyser. Pour plus de détails, on pourra se référer aux discussions du livre *A Wavelet Tour of Signal Processing* [4] autour du principe d'incertitude de Heisenberg.

Pas d'avancement

Le pas d'avancement s permet également de jouer sur la précision temporelle de la TFCT.
Pour des valeurs de s petites devant L, la TFCT permet de donner de meilleures informations sur la localisation temporelle des évènements, mais augmente fortement la dimension de l'espace de représentation du signal. Cette augmentation n'est pas souhaitable car elle entraîne souvent une augmentation trop importante des coûts computationnels, et ce, à la fois lors du calcul de la TFCT mais aussi lors des opérations induites par les algorithmes associés aux différentes tâches que l'on cherche à résoudre. À l'inverse, pour des valeurs de s trop grandes, on perd de l'information sur le signal étudié, ce qui entraîne une baisse importante des performances des algorithmes s'appuyant sur la TFCT. C'est pourquoi on cherchera toujours à faire un compromis entre ces deux bornes. En pratique, on utilise pour l'analyse de signaux audio des valeurs de s comprises dans $[L/4, L/2]$.

2.2.3 Interprétation de la TFCT

La TFCT comme banc de filtres

La TFCT peut être assimilée à l'échantillonnage de plusieurs versions passe-bande du signal. En effet, un filtre s_0 défini par :

$$s_k(t) = w_a(t)e^{-2\pi j\frac{kt}{L}}, \tag{2.18}$$

est une sinusoïde de fréquence k/L modulée par une fenêtre d'analyse w_a. On peut constater que la TFCT de x en (k, n) correspond à la sélection de l'échantillon sn du signal x filtré par s_k avec s le pas d'avancement :

$$\mathrm{TFCT}(n, k) = \big(x \circledast s_k\big)(sn). \tag{2.19}$$

De plus, dans le cas la TFCT, la réponse fréquentielle des différents filtres s_k correspond à un décalage en fréquence de la réponse de la fenêtre w_a. Avec $W_a(f) = \mathcal{F}(w_a)(f)$ on a :

$$S_k(f) = \mathcal{F}(s_k)(f) = W_a(f - k). \tag{2.20}$$

La TFCT est donc une version sous-échantillonnée d'un ensemble de filtres passe-bande ayant des réponses fréquentielles décalées dans le domaine fréquentiel. La figure 2.6 montre ces similarités pour une fenêtre de Hann et deux filtres de fréquences k_1 et k_2.

Discussion

La TFCT est une représentation temps-fréquences, permettant à la fois de capturer l'information fréquentielle sur des petites échelles de temps tout en permettant l'observation des changements de régimes suffisamment lents. Cependant, ces discriminations

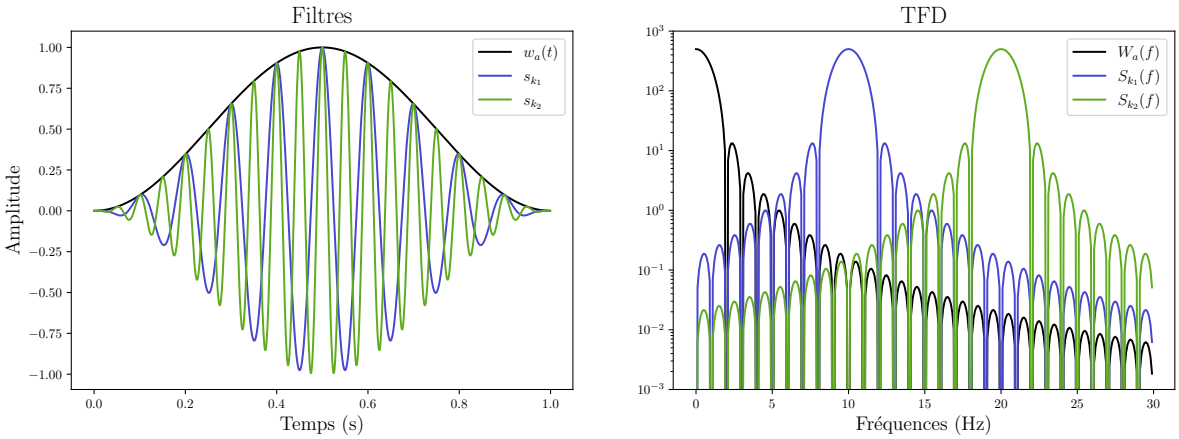


FIGURE 2.6 – (a) Fenêtre de Hann w_a et filtres de fréquences k_1 et k_2 fenêtré par w_a. (b) TFD de ces filtres. On peut voir que la réponse en fréquence des filtres s_{k_1} et s_{k_2} ne sont qu'un décalage en fréquence de la réponse de w_a.

sont effectuées de manière équivalente sur toutes les fréquences. Dans certain cas, le signal ne varie pas à la même vitesse selon les bandes de fréquences étudiées.
Par exemple, on peut adapter la distribution des fréquences analysées afin de coller à la perception humaine comme c'est le cas avec l'échelle des mels [5]. On peut également adapter la taille des fenêtres d'analyse en fonction de la fréquence étudiée, afin de capturer des variations de régimes sur des échelles de temps adaptées aux fréquences analysées. La transformée à Q-constant [6] ou encore la transformée en ondelettes [4] permettent de construire de telles représentations.

2.3 Analyse de la phase dans la TFCT

La phase joue un rôle important dans la perception des signaux audio. Dans la figure 2.4 la phase de la TFCT peut paraître aléatoire. Pourtant, si l'on se munit d'une grille de lecture adaptée, le spectrogramme de phase révèle des structures et des informations sur le signal au même titre que le spectrogramme d'amplitude.

2.3.1 Phase

La phase de la transformée de Fourier discrète d'un signal $\mathcal{F}(x)$ est par définition l'ensemble de ses arguments ϕ :

$$\phi(k) = \arg(\mathcal{F}(x)(k)). \tag{2.21}$$

On peut naturellement étendre cette notion à la TFCT, et construire le spectrogramme de phase ϕ_x tel que :

$$\phi_x(n,k) = \arg(\mathrm{TFCT}_x(n,k)). \tag{2.22}$$

Dans le cas des signaux audio, ajouter une constante à l'ensemble du spectrogramme de phase permet de construire, après application de la TFCT inverse, un nouveau signal audio perçu de manière identique par l'oreille humaine :

$$\phi'_x(n,k) = \phi_x(n,k) + C \tag{2.23}$$

Cette équivalence de perception n'est pas vraie pour des modifications arbitraires N (par exemple, un bruit gaussien) de la phase :

$$\phi'_x(n,k) = \phi_x(n,k) + N(n,k), \tag{2.24}$$

Il existe cependant une large gamme de fonctions ψ telle que la nouvelle phase :

$$\phi'_x(n,k) = \phi_x(n,k) + \psi(n,k), \tag{2.25}$$

n'entraîne pas de modifications dans la perception que l'on a de ce signal.
L'algorithme de reconstruction de phase Griffin-Lim [7] permet de construire des fonctions ψ dans le cas où le spectrogramme d'amplitude est connu. La figure 2.7 présente des portions de signaux dont la phase de la TFCT a été modifiée selon les modalités décrites ci-dessus. À l'oreille, la perception n'est différente que pour le signal dont la phase a été modifiée par l'ajout d'un bruit aléatoire N.

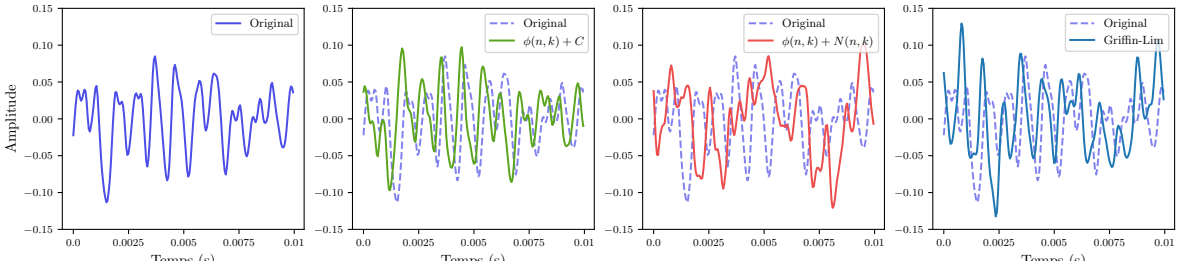


FIGURE 2.7 – Portions de signaux dont la phase de la TFCT a été modifiée.

2.3.2 Dérivées de la phase

Les dérivées partielles de la phase du spectrogramme selon les axes temporel et fréquentiel permettent de faire apparaître des caractéristiques remarquables du signal.

Retard de Groupe

On définit le retard de groupe de la TFCT d'un signal x comme :

$$\mathrm{RD}_x(n,k) = \phi_x(n,k) - \phi_x(n,k-1), \tag{2.26}$$

Le retard de groupe RD correspond à la différence de phases entre les fréquences adjacentes pour chaque trame du spectrogramme de phase.

Pour un signal harmonique et pour un pas d'avancement suffisamment petit, ce retard de groupe sera à peu près constant d'une trame temporelle à une autre. Cette simple transformation permet déjà d'observer certains traits de la phase de la TFCT d'un signal. Dans la figure 2.8, les lignes verticales correspondent à la prononciation de consonnes plausives (\p\, \t\, \k\, ...). En effet, ces dernières étant très localisées dans le domaine temporel, elles "activent" l'ensemble des fréquences lors du calcul de la TFCT. D'une fréquence à une autre, le décalage de phase étant constant, on peut voir apparaître ces lignes verticales puisque le retard de groupe correspond précisément à la localisation temporelle de l'évènement dans la fenêtre.

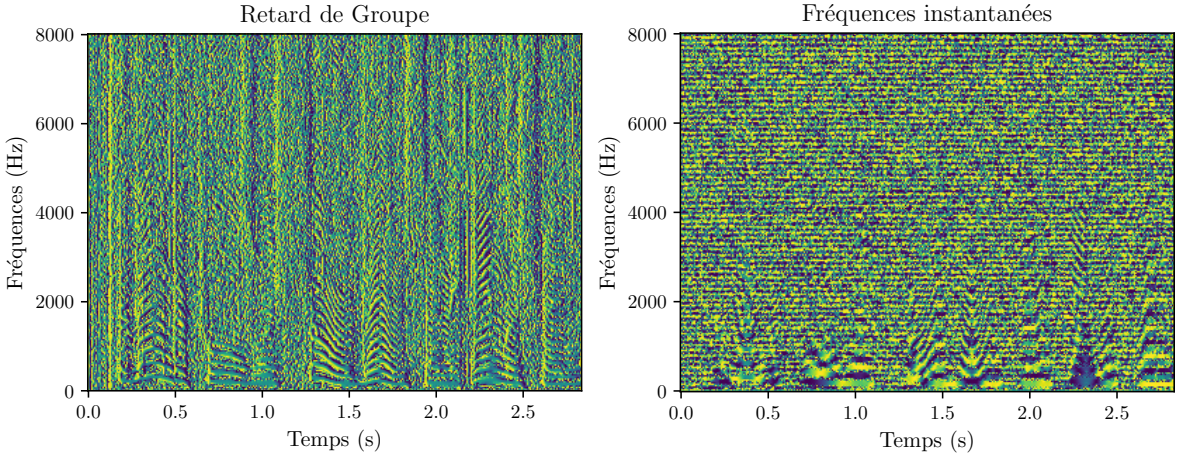


FIGURE 2.8 – Retard de groupe et fréquence instantanée du signal de la figure 2.4. On peut distinguer plus clairement les harmoniques (dans le spectrogramme des fréquences instantanées et des retards de groupes) et les plausives (dans le spectrogramme des retards de groupes) composants un signal de voix.

Fréquence instantanée

De la même manière que l'on a dérivé le spectrogramme de phase selon les fréquences, on peut le dériver dans le temps. On construit ainsi les fréquences instantanées FI [4] :

$$\mathrm{FI}_x(k,n) = \phi_x(n,k) - \phi_x(n-1,k), \tag{2.27}$$

De la même manière que pour le retard de groupe, les signaux harmoniques apparaissent dans cette représentation 2.8. Par exemple, pour un signal harmonique stationnaire, la fréquence instantanée y restera constante sur les bandes de fréquences associées à une sinusoïde.

Discussion

Dans le cas d'un signal harmonique, la perception que l'on se fait du signal est plus ou moins régie dans l'équivalence entre les retards de groupes et les fréquences instantanées des différents signaux. D'une manière plus générale, pour un signal quelconque, on peut dire que la phase de la TFCT d'un signal n'admet pas d'invariances globales, mais une multitude de pseudo-invariances locales. Réussir à reconstruire la phase d'un signal revient donc en partie à reconstruire ces invariances.

2.4 Conclusion

Dans ce chapitre, on a examiné les principes fondamentaux et les méthodes de traitement du signal utiles pour la suite de cette thèse en mettant l'accent sur l'analyse et l'interprétation de la TFCT et de la phase de celle-ci.
On a pu identifier et discuter certaines propriétés des filtres de Fourier utilisés pour le calcul de la TFCT. On a pu aborder l'importance des différents éléments de la TFCT comme le choix de la fenêtre d'analyse afin de minimiser les effets de bords, la taille de la fenêtre afin de trouver un compromis entre discrimination fréquentielle et localisation temporelle.
De plus, on a pu explorer la phase de la TFCT. Malgré son apparence aléatoire, on a pu discuter l'importance de la structure de la phase dans la perception humaine des sons. On a également vu que l'on pouvait faire émerger des structures dans la phase de la TFCT via ses dérivés partielles.
Dans les prochains chapitres, nous explorons plus en détail des techniques de traitement de la phase de la TFCT appliquées à l'apprentissage profond. Nous nous intéresserons aussi à des extensions possibles de la TFCT dans le cas précis des réseaux de neurones profonds appliqués à la séparation de sources audio.

Chapitre 3

Apprentissage profond

Les modèles d'apprentissages pronfonds également appelés réseaux de neurones profonds (DNN pour *Deep Neural Network*) constituent une classe de modèles d'apprentissage automatique privilégiée ces dernières années. En raison de leurs performances, les réseaux de neurones sont devenus une méthode centrale pour une grande majorité des tâches de traitement du signal audio. Un aperçu rapide de ce type d'algorithme est effectué dans ce chapitre, d'abord en présentant brièvement les stratégies d'optimisations mises en jeu dans le cadre des réseaux de neurones puis en présentant les différents composants classiquement utilisés dans ceux-ci. Enfin, on présente également des extensions possibles pour des réseaux de neurones à valeurs complexes qui pourraient a priori être pertinents pour modéliser conjointement les variations d'amplitude et de phase d'un signal.

3.1 Modèle et optimisation

Dans cette thèse, nous nous limitons à l'apprentissage profond dans un contexte *supervisé*, c'est-à-dire que dans la phase d'entraînement, nous supposons que toutes les *données* sont accompagnées d'un *label* constituant la sortie idéale à laquelle l'algorithme devra associer la donnée qui lui est présentée.

3.1.1 Modèle d'apprentissage automatique

Un modèle $f_{\Theta,\Psi}$ est une fonction de $\mathbb{R}^N \to \mathbb{R}^M$ paramétrée par un ensemble Θ de *coefficients* et un ensemble Ψ d'*hyperparamètres*.

3.1.2 Base de données

Dans le cas de l'apprentissage supervisé, on cherche à construire Θ et Ψ tel que :

$$f(x_d|\Theta, \Psi) \approx y_d, \tag{3.1}$$

pour x_d, y_d appartenant à certains ensembles de données. On appelle *base de données*, un ensemble $\mathcal{D}$ constitué de couples $(x_d, y_d) \in \mathbb{R}^N \times \mathbb{R}^M$.

— x_d correspond à une donnée d'entrée ou observation,

— y_d correspond à un label.

Lors de l'apprentissage du modèle, on décompose cette base de données en trois sous-ensembles :

— $\mathcal{D}_{\text{train}}$ utilisé pour optimiser l'ensemble des coefficients Θ de $f_{\Theta,\Psi}$,

— $\mathcal{D}_{\text{validation}}$ utilisé pour suivre l'évolution de l'apprentissage de $f_{\Theta,\Psi}$ sur un ensemble de données indépendant de $\mathcal{D}_{\text{train}}$ et éventuellement pour optimiser les hyper-paramètres Ψ,

— $\mathcal{D}_{\text{test}}$ utilisé après toute optimisation pour qualifier les performances du modèle sur un nouvel ensemble indépendant des deux précédents.

3.1.3 Optimisation

Le choix des paramètres Θ de $f_{\Theta,\Psi}$, se fait par optimisation d'une fonction de coût $\mathcal{L}$:

$$\mathcal{L} : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}, \tag{3.2}$$

$$(y, y') \to \mathcal{L}(y, y'), \tag{3.3}$$

avec $y = y' \implies L(y, y') = 0$.

L'objectif est donc de trouver l'ensemble de paramètres $\tilde{\Theta}$ optimal :

$$\tilde{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \mathcal{L}(f_\Theta(x|\Psi), y). \tag{3.4}$$

Dans le cas des réseaux de neurones, la fonction de coût $\mathcal{L}$ n'est généralement pas convexe par rapport à l'ensemble des paramètres Θ et une procédure d'optimisation de cette fonction conduit en général à un minimum local, qui n'est pas forcément global.

Procédure d'optimisation

Pour les réseaux de neurones, l'algorithme d'optimisation utilisé est la *descente de gradient stochastique* [8] (SGD pour Stochastic Gradient Descent). Cette méthode est itérative. On part d'une valeur initiale de Θ et on la fait évoluer de façon itérative selon la règle de mise à jour suivante :

$$\Theta \leftarrow \Theta + \lambda \frac{\partial \sum_{(x,y)\in\mathcal{B}} \mathcal{L}(f_\Theta(x|\Psi), y)}{\partial\theta}. \tag{3.5}$$

avec $\mathcal{B}$ un sous-ensemble de $\mathcal{D}_{train}$ appelé *batch*. Dans le cas des réseaux de neurones, on utilise l'algorithme de rétropropagation du gradient [9] pour calculer la valeur de $\frac{\partial\mathcal{L}}{\partial\theta}$. Ce problème d'optimisation n'est généralement pas convexe et nous n'avons pas la garantie de convergence vers un minimum local. Afin d'accélérer la convergence, on peut faire usage de méthode utilisant d'autres règles de mise à jour en considérant d'autres critères comme :

— le moment associé aux précédentes mises à jour [10],

— l'adaptation du taux d'apprentissage en fonction des autres mises à jour [11].

Optimisation des hyper-paramètres

Les hyper-paramètres sont optimisés à partir de l'évaluation de $\mathcal{L}(f_\Psi(x|\Theta)$ sur l'ensemble de validation $\mathcal{D}_{val}$ une fois que l'optimisation de Θ pour un Ψ donné a convergé. Ces hyper-paramètres Ψ ne sont généralement pas optimisables à l'aide de l'algorithme de descente de gradient car par exemple, ils appartiennent à un domaine discret. Pour obtenir le meilleur ensemble $\tilde{\Psi}$ d'hyper-paramètres vérifiant :

$$\tilde{\Psi} = \underset{\Psi}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{D}_{\text{val}}} \mathcal{L}(f_\Psi(x|\Theta), y). \tag{3.6}$$

on fait généralement appelle à des heuristiques plus ou moins complexes. Ces algorithmes consistent à apprendre totalement ou partiellement des modèles avec des valeurs de Ψ choisit selon différentes distributions. Par exemple, dans le cas de *Grid Search* [12], il s'agit naïvement d'apprendre des modèles sur toutes les combinaisons associées à une discrétisation finie des différents hyper-paramètres. Cette recherche peut s'avérer très coûteuse, le coût de calcul lors de l'apprentissage des différents modèles testés étant souvent élevé pour les réseaux de neurones.
Afin de minimiser le coût de calcul engendré par la recherche des hyper-paramètres optimaux, des algorithmes comme *HyperBand* [13] proposent de n'apprendre les modèles que partiellement afin d'arrêter l'apprentissage des modèles ayant des performances faibles avant de continuer l'apprentissage sur les hyper-paramétrisations pertinentes.
L'article [14] référence et détaille d'autres algorithmes d'optimisation d'hyper-paramètres couramment utilisés.

3.2 Réseaux de neurones

Les points discutés dans la partie précédente sont valables pour une grande variété d'algorithmes d'apprentissage automatique [15]. Dans cette thèse, on se restreint au cas particulier des réseaux de neurones. Un réseau de neurones consiste en une succession d'opérations simples (linéaires ou non) appliquées à une donnée quelconque afin d'y modéliser des relations potentiellement complexes.

3.2.1 Tenseur

En apprentissage profond, les différentes données circulant dans un réseau de neurones profond sont représentées sous forme de *tenseurs*.
Un tenseur est essentiellement une généralisation des concepts de vecteurs et de matrices à des dimensions supérieures.
Dans le contexte des réseaux de neurones, les tenseurs sont utilisés pour représenter des données, des poids du réseau, des activations intermédiaires, des gradients, des résultats

de prédiction, etc. Ils constituent la structure de base pour les calculs numériques effectués par les modèles d'apprentissage profond.
Un tenseur peut avoir un nombre variable de dimensions, appelées axes ou rangs. Par exemple :

- un tenseur 0D est un scalaire,
- un tenseur 1D est un vecteur,
- un tenseur 2D est une matrice,
- un tenseur 3D est un cube de données, etc...

Chaque dimension d'un tenseur peut avoir une taille arbitraire, ce qui permet de représenter des données complexes. Par exemple, dans un réseau de neurones, un batch d'images est représenté par un tenseur d'ordre 4 dont les dimensions sont notées $B \times C \times H \times W$, avec :

- B la dimension batch qui correspond au nombre d'images différentes dans le batch,
- C, la dimension de canal qui, par exemple, pour une image en couleurs correspond aux trois canaux RGB (Red, Green, Blue),
- H, W sont les dimensions spatiales de hauteur et de largeur.

Dans le cas d'un signal, la représentation pourra être :

- d'ordre 3 et notée $B \times C \times T$, avec B et C le nombre de batchs et de canaux et avec T la dimension temporelle,
- d'ordre 4 et notée $B \times C \times F \times T$, avec F la dimension fréquentielle.

3.2.2 Différentes architectures de réseaux de neurones

L'architecture de réseaux de neurones la plus simple est celle de la classe des perceptrons multicouches [16] (MLP). Un MLP est une fonction de $\mathbb{R}^N \to \mathbb{R}^M$ composée d'une succession de matrices intercalées par des fonctions non-linéaires.
Par exemple, pour un MLP f_Θ à deux couches, le modèle transforme une entrée $x \in \mathbb{R}^N$ en une prédiction $\hat{y} \in \mathbb{R}^M$ de la façon suivante :

$$\hat{y} = \sigma\Big(W_1.\sigma(W_0.x + b_0) + b_1\Big), \tag{3.7}$$

où W_0 et W_1 sont des matrices de poids de taille respective $N \times H$ et $H \times M$, b_0 et b_1 sont des biais et où σ correspond typiquement à une fonction de $\mathbb{R}$ dans $\mathbb{R}$ appliquée indépendamment à toutes les composantes des vecteurs de dimensions quelconques. À partir de cette définition, on peut ajouter autant de couches que nécessaire pour augmenter le pouvoir de représentation du modèle. Généralement, on appelle couche dense une couche respectant les propriétés d'une couche d'un MLP.
Pour des applications de traitement du signal, le MLP présente deux défauts majeurs :

- d'une part, ce type de modèle représente difficilement les structures séquentielles des données,
- et d'autre part, le coût computationnel augmente de manière quasi-cubique avec la dimension de x et la dimension des états intermédiaires. Il est donc impossible

d'utiliser des MLP sur des problèmes pour lesquels la taille de x est trop grande (comme pour les signaux audio où l'ordre de grandeur peut facilement atteindre la centaine de milliers d'échantillons).

Pour pallier ces limitations, on utilise généralement des modèles plus structurés.

Réseaux de neurones récurrents

Un réseau de neurones récurrent (ou RNN pour *Recurrent Neural Network*) est un type de réseau de neurones spécialement conçu pour traiter des données séquentielles. Contrairement au MLP, les RNNs exploitent la dépendance séquentielle entre les données. Dans le cas d'une séquence de données $x \in \mathbb{R}^{N \times T}$, un RNN traitera les différents éléments $x_t \in \mathbb{R}^N$ récursivement :

$$\text{RNN}(x_{t+1}) = h_{t+1} = \sigma(W.[x_{t+1} : h_t] + b), \tag{3.8}$$

où h_t est un vecteur de $\mathbb{R}^H$ appelé état interne, où W est une matrice de poids de dimension $\mathbb{R}^{C+H,H}$, et où $[. : .]$ correspond à une opération de concaténation.
Un RNN consiste donc en l'application de la même couche dense sur l'ensemble des éléments de la séquence d'entrée. L'état interne h_t permet de garder en mémoire les informations induites par les t premiers éléments de la séquence x.
Les RNNs peinent cependant à conserver les informations à mesure que l'on avance dans la séquence. Ils sont limités d'abord par leur pouvoir de représentation, mais également par des problèmes d'optimisation, induite par la rétropropagation du gradient :

- l'évaporation du gradient quand la norme de celui-ci tend vers 0 avec l'éloignement temporel,
- et inversement l'explosion du gradient quand la norme tend vers ∞.

Il existe de nombreuses améliorations sur les RNNs permettant de résoudre ou de contrôler ces problèmes de normes lors de l'optimisation. On peut notamment citer les *Long Short Term Memory* (LSTM) [17], les *Gated Recurrent Unit* (GRU) [18] ou les RNN unitaires [19].

Réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels (ou CNN pour *Convolutional Neural Networks*) sont des modèles dont les différentes couches permettent d'ajouter de la structure dans les opérations que l'on applique à la donnée d'entrée en mimant d'un certain point de vue l'opération de convolution du traitement du signal définie dans l'équation 2.6.
Pour une donnée bidimensionelle $x \in \mathbb{R}^{C_1,T}$ et un tenseur de poids $W \in \mathbb{R}^{C_1,C_2,K}$ la convolution 1D de x par W est définie telle que :

$$(x \circledast W)[t] = \sum_{k=0}^{K-1} x[t+k].W[k], \quad t = 0, 1, \ldots, T-1 \tag{3.9}$$

ici, $(x \circledast W)$ correspond à un tenseur de dimensions $C_2 \times T$. De la même manière que la TFCT, on peut modifier la dimension temporelle de $x \circledast W$ en ajoutant un pas d'avancement ou *stride* P (définie dans le paragraphe 2.2.2) tel que :

$$(x \circledast W)[t] = \sum_{k=0}^{K-1} x[t+k].W[k], \quad t = 0, P, 2P, \ldots \tag{3.10}$$

cette fois, $(x \circledast W)$ correspond à un tenseur de dimensions $C_2 \times T/P$. On peut d'ailleurs noter qu'en associant à x un signal temporel de $\mathbb{R}^{1\times T}$ et à W la matrice associée à la transformée de Fourier discrète de l'équation 2.5 redimensionnée en un tenseur de $\mathbb{C}^{1\times K\times K}$ on retrouve la définition de la TFCT.
Les couches de convolutions permettent à la fois de traiter la donnée d'entrée de manière locale et d'adapter la dimension temporelle par l'utilisation d'un pas d'avancement.
Cette définition de la convolution s'étend aisément pour des données spatiales (comme des images) et est à la base de nombreuses avancées dans le domaine de l'apprentissage profond [20, 21].

Correspondance avec le traitement du signal

Comme on l'a vu précédemment, on peut rapprocher les couches de convolutions à des opérations de filtrages.
On peut également considérer les RNNs comme une généralisation des filtres auto-régressifs [22] (AR), où l'on estime la suite d'une séquence à partir des éléments précédents. Une des problématiques de cette thèse sera d'évaluer cette correspondance entre le domaine du traitement du signal et le domaine de l'apprentissage profond pour les couches de convolutions.

3.2.3 Autres composantes

Il existe deux autres composantes essentielles au bon fonctionnement des DNNs : les fonctions de non-linéarités et les fonctions de normalisations.

Non-linéarités

Dans un DNN, on appelle non-linéarités des fonctions non-affines de $\mathbb{R}$ dans $\mathbb{R}$ appliquées terme à terme à toutes les composantes d'un tenseur de dimension quelconque. L'objectif de ces fonctions est de casser les relations de linéarités entre les différentes couches du modèle afin d'accroitre les capacités de représentation. Une non-linéarité se doit d'être différentiable presque partout afin de permettre la rétro propagation du gradient lors de l'optimisation. La fonction ReLU (pour *Rectified Linear Unit*) est l'une des fonctions les plus utilisées dans les DNN. Elle est définie telle que :

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{sinon,} \end{cases} \tag{3.11}$$

D'autres non-linéarités comme la fonction sigmoïde σ :

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{3.12}$$

ou la fonction arctangente peuvent être utilisées si l'on veut forcer une certaine structure sur la sortie : cependant, leurs dérivées respectives tendent vers 0 à mesure que la norme de x grandit, rendant leur pertinence limitée pour des modèles très profonds.

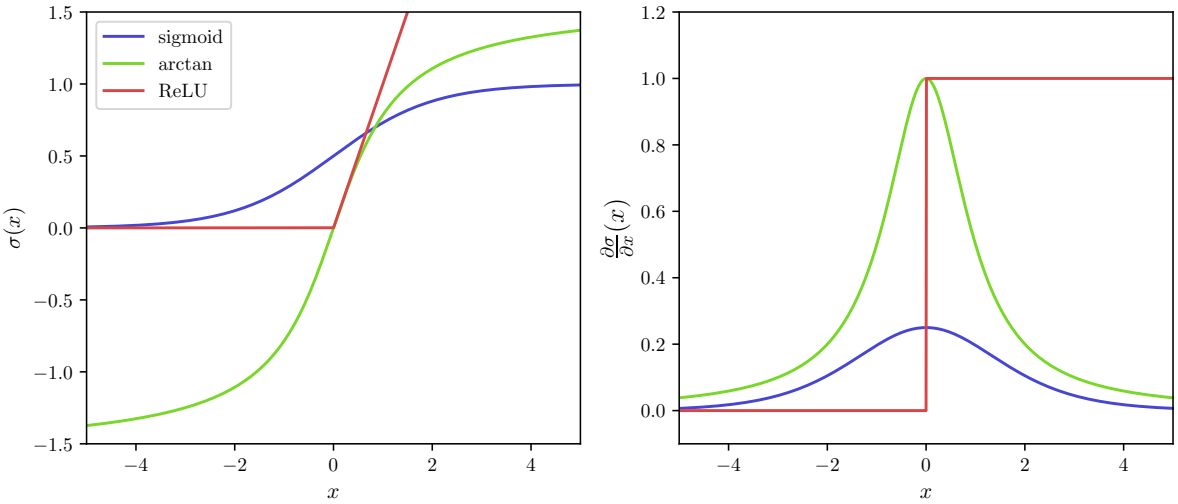


FIGURE 3.1 – Fonctions de non-linéarité classiques et leurs dérivées respectives.

Couches de normalisation

Les fonctions de normalisation permettent de corriger la moyenne et la variance d'un ensemble de données entre les différentes couches linéaires.
Une fonction de normalisation couramment utilisée est la Batch-Normalisation [23] et est décrite comme suit. Etant donné un tenseur $x \in \mathbb{R}^{B \times C \times T}$, où B, C, T sont des dimensions correspondant respectivement à la taille du batch, le nombre de canaux et la longueur de la séquence, on peut calculer les moyennes et variances suivantes pour chaque canal $C \in [0, C-1]$:

$$\mu_c = \frac{1}{BT} \sum_{b,t} x_c^{b,t}, \tag{3.13}$$

$$\sigma_c = \frac{1}{BT} \sum_{b,t} (x_c^{b,t} - \mu_c)^2, \tag{3.14}$$

les tenseurs μ et σ sont donc des vecteurs de $\mathbb{R}^C$. La Batch-Normalisation de x est finalement calculée de la manière suivante :

$$\tilde{x}_c^{b,t} = \frac{x_c^{b,t} - \mu_c}{\sigma_c + \epsilon}, \tag{3.15}$$

$$\mathrm{BN}(x_c^{b,t}) = \gamma_c \tilde{x}_c^{b,t} + \beta_c, \tag{3.16}$$

où les vecteurs $\gamma \in \mathbb{R}^C$ et $\beta_c \in \mathbb{R}^C$ sont des paramètres optimisables permettant d'appliquer une transformation affine sur les données après avoir effectué la normalisation $\tilde{x}$. Le choix des différents axes sur lesquels on décide de centrer-réduire les données permettent de construire des fonctions de normalisations différentes. En considérant ce même vecteur x on peut ainsi définir d'autres normalisations communément utilisées :

— LayerNorm [24] : Normalisation sur les axes des canaux et du temps,
— InstanceNorm [25] : Normalisation sur le seul axe du temps.

L'utilisation de ces fonctions est essentielle pour accélérer et aider la convergence des modèles. Cependant, les raisons expliquant le rôle exact des normalisations dans les réseaux de neurones profonds sont encore mal comprises.

3.3 Réseaux de neurones à valeurs complexes

En s'intéressant à la TFCT d'un signal, on peut naturellement être amené à traiter celle-ci directement dans le domaine des nombres complexes à l'aide d'un DNN à valeurs complexes. Cependant, certaines des définitions présentées dans les parties précédentes ne s'étendent pas trivialement dans le domaine complexe. L'objectif de cette partie est donc de présenter succinctement les principales méthodes utilisées pour définir et optimiser des réseaux de neurones à valeurs complexes.

3.3.1 Optimisation

Optimiser un modèle à valeurs complexes quelconque revient à optimiser une fonction d'un espace vectoriel sur $\mathbb{C}$ vers $\mathbb{R}$. Pour utiliser un algorithme de type descente de gradient (voir section 3.1.3), il faut préciser en quoi consiste une dérivée par rapport à une variable complexe.

Fonctions holomorphes

Une fonction complexe f est dite holomorphe [26] sur un domaine Ω inclus dans $\mathbb{C}$ si la limite suivante est définie pour tout $z_0 \in \Omega$:

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}. \tag{3.17}$$

Les fonctions holomorphes sur $\mathbb{C}$ sont également appelées fonctions entières et sont développables en série entière autour de tout $z_0 \in \mathbb{C}$:

$$f(z) = \sum_{n=0}^{+\infty} a_n (z - z_0)^n, \text{ avec } a_n = \frac{f^{(n)}(z)}{n!}. \tag{3.18}$$

Ainsi, les polynômes et en particulier les fonctions linéaires sont des fonctions holomorphes. L'intérêt principal dans l'utilisation de fonctions holomorphes est qu'elles garantissent de déformer l'espace de manière très régulière. En particulier, appliquer de telles fonctions sur des points de la TFCT (dont la phase est, comme on l'a vu dans le paragraphe 2.3.1, relative) garantirait une invariance locale aux rotations induites par f sur la phase de la TFCT.
Cependant, l'optimisation sur des fonctions holomorphes rencontre une difficulté majeure : une fonction holomorphe n'est bornée que si elle est constante. Deux cas de figure se posent alors :

— la fonction f diverge pour $|z| \to +\infty$,
— et/ou la fonction f présente des singularités [1] dans le plan complexe.

Cette propriété rend l'optimisation de fonctions holomorphes instable, car il est impossible de prévoir les instabilités numériques induites par ces valeurs.

Fonctions complexes quelconques

La différentiabilité au sens de Wirtinger [27], donne un cadre formel au calcul de gradient pour les fonctions à valeurs complexes. On ne s'étendra pas beaucoup plus sur les simplifications s'opérant lors du calcul des gradients mais il est à noter que les calculs sont strictement équivalents à l'analogie avec des fonctions réelles de $\mathbb{R}^2$.
Les plateformes logicielles de réseaux de neurones comme pyTorch ou Tensorflow conviennent donc parfaitement pour implémenter des fonctions à valeurs complexes. Pour plus de détails sur les simplifications mises en jeu, on peut se référer à la thèse de Andy Saroff [28].

3.3.2 Architectures complexes

Présentons maintenant les différences entre les blocs des réseaux de neurones à valeurs complexes et des réseaux de neurones à valeurs réelles couramment utilisés dans la littérature récente.

Couches linéaires

Pour transformer une couche linéaire réelle en une couche linéaire complexe, il suffit de passer les poids réels des tenseurs associés à des poids complexes. Par exemple, pour une matrice $W \in \mathbb{R}^{N \times M}$, il suffit de construire la matrice $W' \in \mathbb{C}^{N \times M}$ se décomposant

1. si z_0 est un point singulier de f alors la norme de z tendra vers $+\infty$ à mesure que z tend vers z_0

donc en une partie réelle et une partie imaginaire :

$$W' = W'_{\Re} + iW'_{\Im}, \tag{3.19}$$

à dimensions équivalentes, le passage en poids complexes multiplie par deux le nombre de paramètres et par quatre le nombre d'opérations. En effet, on a pour un vecteur $x \in \mathbb{C}^N$:

$$W'x = \Big(W'_{\Re}\Re(x) - W'_{\Im}\Im(x)\Big) + i\Big(W'_{\Im}\Re(x) + W'_{\Re}\Im(x)\Big). \tag{3.20}$$

Couches convolutionnelles

De la même manière que pour les couches linéaires, les couches convolutionnelles complexes sont définies à l'aide de tenseurs de poids complexes. La convolution 1D d'une donnée $x \in \mathbb{C}^{C_1,T}$ par un tenseur complexe de poids $W \in \mathbb{R}^{C_1,C_2,K}$ se résume en l'application de quatre convolutions réelles :

$$x' = (x \circledast_{\mathbb{C}} W) = \Big(\Re(x) \circledast \Re(W) - \Im(x) \circledast \Im(W)\Big) + i\Big(\Re(x) \circledast \Im(W) + \Im(x) \circledast \Re(W)\Big), \tag{3.21}$$

avec $x' \in \mathbb{C}^{C_2,\frac{T}{p}}$ avec p le pas d'avancement de la convolution.

Couches récurrentes

L'application d'une couche récurrente est essentiellement l'application d'une couche linéaire appliquée récursivement sur une donnée temporelle, ainsi l'extension aux réseaux complexes est immédiat. Cependant, faire ainsi demanderait de modifier en profondeur les cellules définies dans les librairies de calculs comme PyTorch et TensorFlow. Une définition plus simple à mettre en oeuvre est donnée par exemple dans [29], qui propose de définir un RNN complexe comme l'application de plusieurs RNN réels sur les parties réelles et imaginaires indépendamment tel que pour une donnée $x \in \mathbb{C}^{C,T}$ et un RNN complexe $\mathrm{RNN}_{\mathbb{C}}$ on ait :

$$\mathrm{RNN}_{\mathbb{C}}(x) = \Big(\mathrm{RNN}_{\Re}(\Re(x)) - \mathrm{RNN}_{\Im}(\Im(x))\Big) + i\Big(\mathrm{RNN}_{\Im}(\Re(x)) + \mathrm{RNN}_{\Re}(\Im(x))\Big), \tag{3.22}$$

où $\mathrm{RNN}_{\Im}$ et $\mathrm{RNN}_{\Re}$ sont des RNN réels. Comme ces réseaux ne font pas véritablement intervenir les propriétés complexes des données manipulées, on parlera plutôt de *Split-RNN*.

Entre une architecture réelle et une architecture complexe, il faudra toujours veiller à comparer le coût computationnel pour obtenir des comparaisons équitables entre des modèles réelles et des modèles complexes.

3.3.3 Autres composantes

Non-linéarités

Les non-linéarités dans les réseaux de neurones à valeurs complexes jouent un rôle particulièrement important dans l'interprétation que l'on peut se faire sur l'espace de ses représentations.
Dans la pratique, les non-linéarités complexes utilisées dans les réseaux de neurones sont des fonctions non-holomorphes. On peut les décomposer en deux catégories :

— Les fonctions σ traitant un nombre z sur la phase et l'amplitude de manière indépendante, où $\sigma(z) = \sigma_{\text{Amp}}(|z|)e^{i\sigma_\phi(\arg(z))}$ qui créent une distorsion contrôlée dans la phase de z (potentiellement nulle si σ_ϕ correspond à l'identité),
— Les fonctions σ traitant les parties réelle et imaginaire de z indépendamment, tel que $\sigma(z) = \sigma_\Re(\Re(z)) + i\sigma_\Im(\Im(z))$.

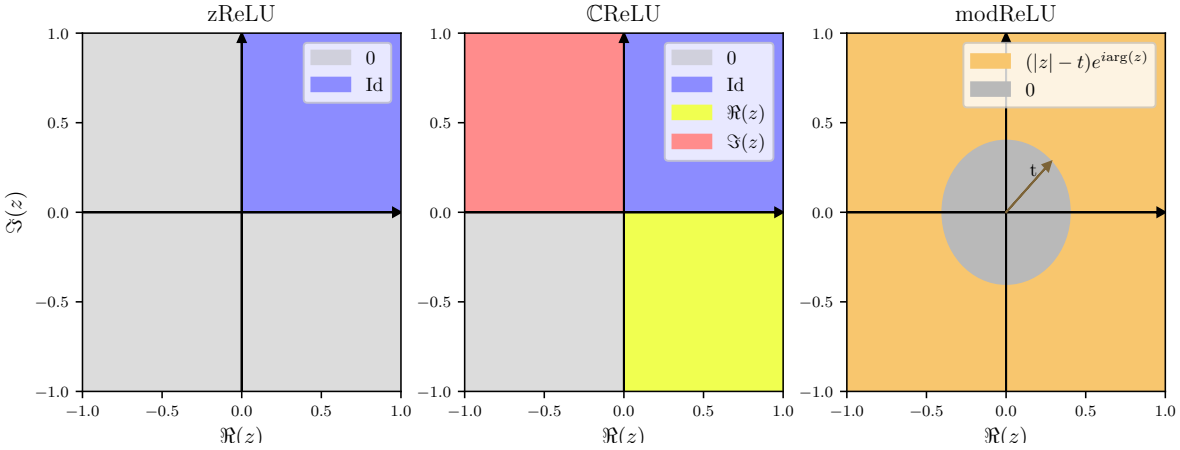


FIGURE 3.2 – Non-linéarité complexe

Dans la première catégorie, on utilise principalement les fonctions zReLU et modReLU [19] :

$$\text{zReLU}(z) = \begin{cases} z & \text{si } z \bmod 2\pi \in [0, \pi/2], \\ 0 & \text{sinon,} \end{cases} \tag{3.23}$$

$$\text{modReLU}(z|\tau) = \text{ReLU}(|z| - \tau)e^{i\arg(z)}. \tag{3.24}$$

$$\tag{3.25}$$

Dans la seconde, on retrouve la non-linéarité la plus couramment utilisée, ℂReLU, qui amène expérimentalement aux meilleures performances :

$$\mathbb{C}\text{ReLU}(z) = \text{ReLU}(\Re(z)) + i\text{ReLU}(\Im(z)). \tag{3.26}$$

La figure 3.2 donne un aperçu de l'image du plan complexe par ces trois non-linéarités.

Normalisation complexe

L'application de la normalisation sur des tenseurs à valeurs complexes n'est pas immédiat. En effet, si on considère les parties réelles et imaginaires indépendamment la covariance entre celles-ci n'est pas forcément nulle. Dans l'article [30], Trabelsi et al. note qu'expérimentalement, la covariance des parties réelles et imaginaires des différentes représentations augmentent à mesure que l'on avance dans le modèle.
Cette augmentation de la covariance n'est pas désirable, car elle implique que les parties réelles et imaginaires deviennent redondante.
Pour assurer d'avoir une covariance stable, on blanchit la représentation complexe. En omettant l'indiçage des équations 3.13 et 3.15, la batch normalisation complexe devient [30] :

$$\tilde{z} = \mathbb{V}^{-\frac{1}{2}}(z - \mathbb{E}(z)), \tag{3.27}$$

$$\mathbb{C}\text{BatchNorm}(z) = \gamma z + \beta. \tag{3.28}$$

$$\tag{3.29}$$

Avec $\mathbb{V}$ la matrice de covariance suivante :

$$\mathbb{V} = \begin{pmatrix} \text{Cov}(\Re(z), \Re(z)) & \text{Cov}(\Re(z), \Im(z)) \\ \text{Cov}(\Im(z), \Re(z)) & \text{Cov}(\Im(z), \Im(z)) \end{pmatrix}. \tag{3.30}$$

Même si on résout effectivement le problème de corrélation pour les nombres complexes, et de la même manière que pour les non-linéarités, les distorsions induitent par cette transformation ne sont pas forcément désirables pour modéliser correctement la phase.

Discussion

Finalement, les réseaux de neurones à valeurs complexes utilisés tel qu'aujourd'hui souffrent d'un manque d'interprétabilité encore plus important que pour leurs analogues à valeurs réelles. Les différentes méthodes, qui comme on l'a vu, sont peu satisfaisantes, rendent l'interprétation des différents états latents des modèles complexes encore moins intuitifs que dans le cas réel. À l'inverse, l'utilisation de fonctions interprétables d'un point de vue complexe rendent l'optimisation impossible. La pertinence de ce type de modèle reste donc a priori assez limitée dans l'état actuel de la recherche.

3.4 Conclusion

Dans ce chapitre, on a inventorié les principales stratégies d'optimisation et les principales architectures des réseaux de neurones profonds.
Dans la suite de cette thèse, les procédures d'optimisations utilisées feront référence aux méthodes d'optimisation des paramètres et d'hyperparamètres présentées ici.
On a pu voir que les couches de convolutions et les couches récurrentes peuvent être assimilées à des opérations classiques de traitement du signal, respectivement le filtrage

linéaire et le filtrage auto-régressif. Dans le cas des convolutions, cette remarque sera la base des chapitres 6 et 7 de cette thèse.
Enfin, on a présenté les réseaux de neurones à valeurs complexes par analogie de leurs équivalents réels. On a brièvement pu discuter de la pertinence des réseaux de neurones à valeurs complexes dans l'état actuel des choses. On verra dans le chapitre 5 des cas particuliers de ceux-ci.

Chapitre 4

Séparation de sources monaurale et rehaussement de la parole

La phase d'un signal audio est très peu informative pour une tâche telle que la classification de signal ou l'identification de locuteur. En revanche, la phase contient de l'information primordiale pour des tâches de reconstruction ou de génération de signal audio comme le rehaussement de la parole, la séparation de sources, la compression ou la synthèse vocale. Une approche par apprentissage profond sur ce type de tâche doit donc avoir la capacité de modéliser la phase d'un signal.
Dans ce chapitre, on se concentre sur la tâche de séparation de sources [31] et nous présentons les principales méthodes à base de réseaux de neurones pour répondre à cette problématique.

4.1 Modélisation du problème

La séparation de sources consiste en général à estimer plusieurs signaux d'intérêt à partir de l'observation de un ou plusieurs signaux appelés *mélanges*. Dans le cas de *mélanges linéaires*, on observe R signaux, $x_0(t), ..., x_{R-1}(t)$ avec :

$$\forall r,\ x_r(t) = \sum_{q=0}^{Q-1} \alpha_{q,r} s_q(t - \delta_{q,r}), \tag{4.1}$$

où :
- $s_0, ..., s_{Q-1}$ sont les signaux d'intérêt (ou sources),
- $\forall (q, r) \in [\![0, Q-1]\!] \times [\![0, R-1]\!]$, $\alpha_{q,r}$ correspond au facteur d'échelle du signal q sur l'observation r et $\delta_{q,r}$ correspond à un retard.

Le problème de séparation de sources peut être défini dans un cadre multicanal (avec l'utilisation de multiples microphones ou antennes) et sur-determiné (avec un nombre de sources strictement inférieur au nombre de microphones utilisés).

Les solutions à ce problème font usage des retards relatifs des différents signaux sur les différents capteurs pour spatialiser les signaux et les isoler selon leur direction d'arrivée.

4.1.1 Séparation de sources monaurale

Quand le nombre de capteurs est égal à 1, on parle de séparation de sources monaurale. Dans ce contexte, le problème est sous-déterminé : il y a plus d'inconnues que d'équations et il ne peut être possible de résoudre le problème qu'en injectant de l'information a priori sur les signaux en présence.
Plus formellement, on observe un mélange linéaire x unique composé d'un nombre Q de sources s_q :

$$x(t) = \sum_{q=0}^{Q-1} s_q(t) + n(t), \tag{4.2}$$

où $n(t)$ correspond à un signal sonore quelconque que l'on ne cherche pas à reconstruire. Ce signal n peut correspondre à un ensemble de sources audio indésirables et/ou à des artefacts liés à la qualité de l'enregistrement.
Idéalement, on cherche à construire une fonction de reconstruction f, tel que :

$$f(x)(t) = [s_0(t), ..., s_{Q-1}(t)]. \tag{4.3}$$

La difficulté de cette tâche réside dans le fait que les différents signaux peuvent interférer entre eux. Si les signaux sources ont des supports fréquentiels disjoints, il suffit d'appliquer des filtres passe-bande bien choisis afin d'extraire les différentes sources. En effet, étant donnés une certaine plage de fréquences et une certaine plage temporelle, l'addition de deux sources peut entraîner des effets de suppressions ou d'amplifications sur certaines zones temps-fréquences analysées.
Les informations de phase et d'amplitude des différentes sources peuvent cependant être distillées dans des réseaux de neurones profonds dans la phase d'apprentissage. C'est pourquoi ils constituent aujourd'hui l'état de l'art au problème.

4.2 Approches par réseaux de neurones profonds

Les algorithmes d'apprentissage profond permettent d'intégrer un a priori fort et élaboré sur les données. C'est pourquoi ils constituent une classe d'approches adéquate pour traiter le problème de séparation et forment aujourd'hui la majorité des approches de l'état de l'art. Les premiers algorithmes efficaces d'apprentissage profond effectifs apparaissent avec les travaux de Huang et al [32]. Depuis, les performances de ces derniers ont considérablement été améliorées pour les tâches de séparation de sources et de rehaussement de la parole [31]. Il existe plusieurs types de stratégies pour résoudre la tâche de séparation de sources :

— Les méthodes de masquages qui cherchent à reconstruire un masque à appliquer sur une projection du mélange afin de retrouver les différentes sources. Cette stratégie est la plus largement développée dans la communauté. Son principe est détaillé plus finement dans la section suivante.

— Les méthodes bout-en-bout qui cherchent à reconstruire le signal directement dans le domaine temporel sans stratégie de masquage. Ce type de modèle se résume à des architectures particulières comme Wav-Net [33, 34] ou Wav-UNet [35] (détaillée précisément dans le chapitre 7) qui permettent d'encoder puis de régénérer le signal au niveau de l'échantillon. D'abord popularisées par des méthodes génératives comme le modèle SEGAN [36] basé sur une méthode d'optimisation adversaire [37], les méthodes bout-en-bout ont regagné un certain intérêt depuis l'essor des méthodes de diffusion [38] appliquées à la séparation de sources [39].
— Les méthodes dites DDSP pour *Differentiable Digital Signal Processing* reposent quant à elles sur des modèles de synthèse sonore comme les méthodes sommant sinusoïdes et bruits [40]. Les réseaux de neurones associés à ce type de méthode ne cherchent pas à reconstruire explicitement le signal, mais simplement à estimer des paramètres d'un modèle de production de parole classique, comme un modèle source-filtre dans le cas de la séparation de sources par synthèse vocale [41].

4.2.1 Approches par estimation de masques

Pour résoudre le problème de séparation de sources monaurale, une classe d'algorithmes cherche à estimer des masques à appliquer à la TFCT [42].
Plus formellement, étant donné la TFCT de x, $X \in \mathbb{C}^{F\times T}$, on cherche à construire Q matrices $M_q \in \mathbb{C}^{F\times T}$ telles que :

$$\hat{S}_q(f,t) = X(f,t) \odot M_q(f,t), \tag{4.4}$$

où $\hat{S}_q$ correspond à une estimation de la TFCT de s_q et où $\odot$ correspond au produit terme à terme entre deux tenseurs de dimensions compatibles. Dans la pratique, on estime un $\hat{S}_q$ aussi proche de S_q que possible afin de minimiser les distorsions de perception de $\hat{s}_q$. On estime ensuite $_q$ par $\hat{s}_q = \text{TFCT}^{-1}(\hat{S}_q)$. Dans cette partie, on se place dans un cas oracle, c'est-à-dire que l'on connaît a priori les signaux sources à estimer.

Masques d'amplitude idéaux

On peut construire une première famille de masque M_q réels et positifs afin de corriger l'amplitude du mélange. Ainsi, on estime :

$$\hat{S}_q(f,t) = \Big(M_q(f,t) \odot A_x(f,t)\Big) e^{j\phi_x(f,t)}, \tag{4.5}$$

où A_x et ϕ_x correspondent respectivement au spectrogramme d'amplitude et de phase de x. Dans cette configuration, la phase du mélange est donc utilisée pour estimer la phase des signaux.

Pour une source q d'un mélange x, les masques d'amplitude les plus couramment utilisés sont :

$$\text{Le Masque Idéal Binaire} \qquad \mathrm{IBM}_q(f,t) = \begin{cases} 1 & \text{si } A_q(f,t) = \max_{q' \in [\![1,Q]\!]} A_{q'}(f,t), \\ 0 & \text{sinon,} \end{cases} \tag{4.6}$$

$$\text{Le Masque de Ratio Idéal} \qquad \mathrm{IRM}_q(f,t) = \frac{A_q}{\sum_{q' \in [\![0,Q-1]\!]} A_r}, \tag{4.7}$$

$$\text{Le Masque de Wiener Idéal} \qquad \mathrm{WFM}_q(f,t) = \frac{A_q^2}{\sum_{q' \in [\![0,Q-1]\!]} A_{q'}^2}. \tag{4.8}$$

La figure 4.1 présente une visualisation de ces différents masques dans un cas où x correspond au mélange de deux voix.

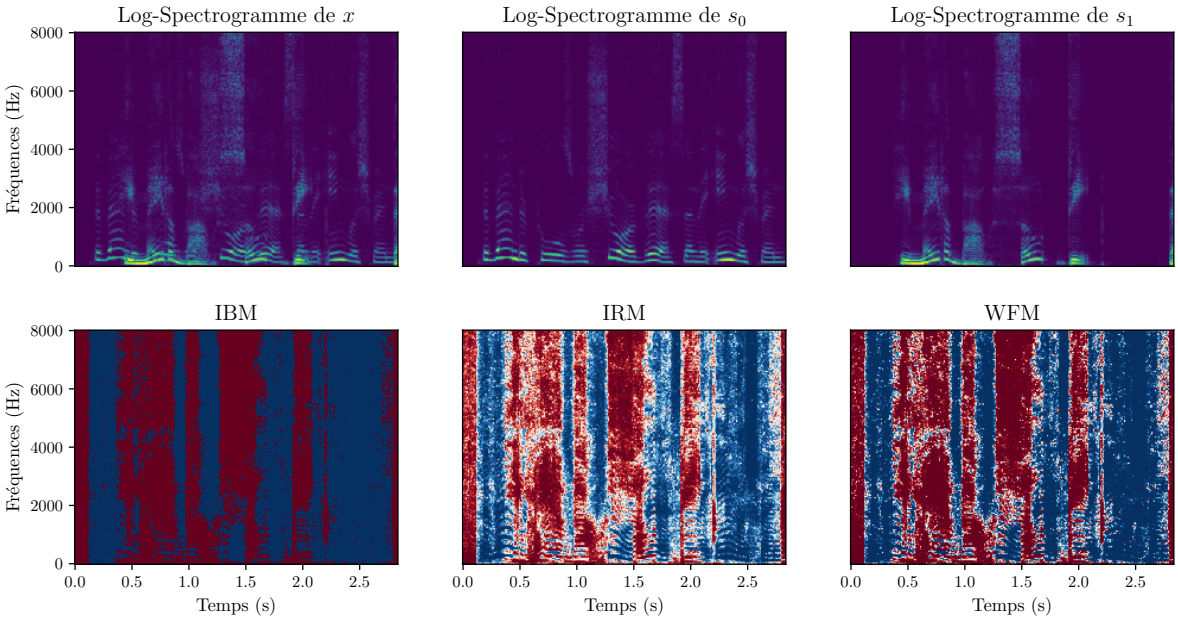


FIGURE 4.1 – (Haut) Spectrogramme de Log-Amplitude d'un mélange de deux voix $x = s_0 + s_1$. Spectrogramme des voix s_0 et s_1. (Bas) Masques Idéaux des différentes voix, en rouge voix 1, en bleu voix 2 et en blanc autant des deux.

Le problème majeur des masques d'amplitude est qu'ils ne prennent pas en compte la correction de phase à apporter pour estimer au mieux les $\hat{S}_q$. En effet, dans le cas où les points temps-fréquences des différentes sources se superposent, des erreurs peuvent apparaître et induire de lourdes distorsions dans la perception du signal reconstruit. C'est pour pallier ce problème que l'estimation de masques prenant en compte la phase peut avoir un intérêt.

Masques Amplitude-Phase

Afin de prendre en compte la phase, plusieurs stratégies de masquage peuvent être mises en place. On peut simplement utiliser le masque idéal complexe qui permet de reconstruire parfaitement le signal, $\mathrm{CM}_q \in \mathbb{C}^{F,T}$ avec :

$$\mathrm{CM}_q(f,t) = \frac{S_q(f,t)}{X(f,t)}. \tag{4.9}$$

On peut également utiliser des masques prenant en compte la différence des phases entre les TFCT des sources et du mélanges tout en restant dans le domaine réel. On construit ces masques *Phase-Sensitive Filter* [43] (PSF) tel que :

$$\mathrm{PSF}_q = \frac{A_q}{A_x}\cos(\phi_q - \phi_x) = \Re(\mathrm{CM}_q(f,t)). \tag{4.10}$$

Masque de phase

On peut également se concentrer à ne modifier qu'exclusivement la phase pour séparer les sources. Même si ce type de stratégies est moins efficace que les stratégies à base d'amplitude, il est bon de noter qu'un travail exclusif sur la phase permet d'améliorer la séparation :

— dans l'article [44], les auteurs montrent qu'en appliquant un spectrogramme de phase bien choisi de la source cible (fenêtre de Chebyshev) sur le spectrogramme d'amplitude du mélange, on arrive à obtenir une restauration des signaux sources de qualité significativement intéressante.

— à l'inverse, si on connaît le spectrogramme d'amplitude du bruit, on peut corriger la phase du mélange à partir de celui-ci afin de restaurer la voix [45] après application de la TCFT inverse.

Comparaison des performances oracles des masques temps-fréquences

Sur une problématique de rehaussement de parole, on peut observer sur la table 4.1 que les filtres PSF obtiennent les meilleures performances de reconstruction. On peut noter que le masque de phase pure en utilisant un fenêtrage de Chebyshev permet d'obtenir un gain de performance significatif malgré une modification nulle de l'amplitude.

	Mélange	IBM	IRM	WFM	PSF	Chebyshev
SDR	1.19	15.66	15.21	17.31	20.61	8.81
PESQ	1.60	3.22	3.72	3.85	4.10	2.52
STOI	0.74	0.89	0.96	0.96	0.97	0.86

TABLE 4.1 – SDR, PESQ et STOI après reconstruction des sources sur les différents masques présentées jusqu'ici pour un problème de rehaussement de la parole. Ces trois métriques sont décrites plus précisément dans la partie 4.3

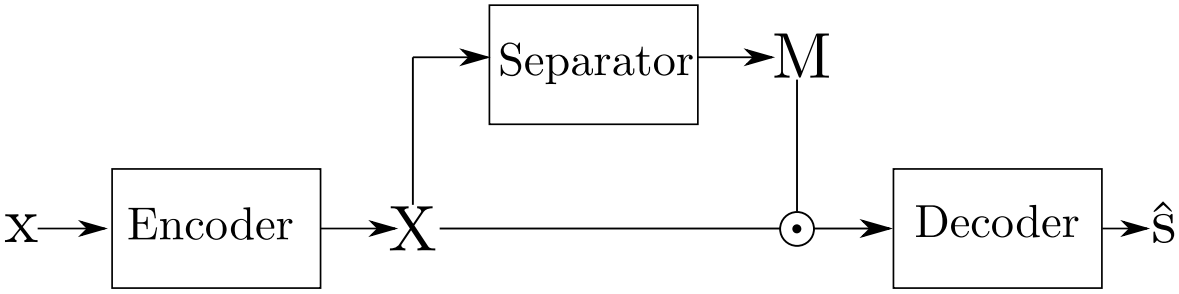


FIGURE 4.2 – Schéma d'une architecture de masquage.

4.2.2 Modèles de masquage en apprentissage profond

Les modèles de masquage cherchent à optimiser un masque à appliquer sur une représentation 2D du signal comme on l'a décrit dans la partie précédente. Pour ça, on peut choisir d'entraîner les modèles à estimer un certain type de masque idéal [46, 47] (comme ceux présentés dans la section précédente) ou bien d'entraîner le modèle à estimer le signal directement dans le domaine temporel mais en conservant le principe de la multiplication d'un masque donné à une représentation latente bi-dimensionelle [48, 49].

Stratégie globale

Les modèles de masquage sont composés de trois parties :

— Une première partie, appelée encodeur, consistant à appliquer une fonction f_e pour projeter le mélange $x \in \mathbb{R}^T$ dans un espace bi-dimensionelle de dimensions $C \times T'$. C correspond à la dimension des canaux et T' correspond à la dimension temporelle :

$$X = f_e(x),\ X \in \mathbb{R}^{C,T'}. \tag{4.11}$$

— Une deuxième partie, appelée séparateur f_s, consistant à construire Q masques M_q de dimensions $C \times T'$:

$$M_q = f_s(X),\ M_q \in \mathbb{R}^{C,T'},\ q \in [1, Q]. \tag{4.12}$$

— Une dernière partie, appelée décodeur f_d, consistant à reconstruire les sources s_q dans le domaine temporel après avoir pris le produit terme à terme des différents M_q masques par la projection X :

$$\hat{s}_q = f_d(X \odot M_q),\ \hat{s}_q \in \mathbb{R}^T. \tag{4.13}$$

L'ensemble de l'algorithme peut donc se résumer à l'équation suivante :

$$\hat{s}_q = f_d\Big(f_m \odot \big(f_e(x)\big)_q \cdot f_e(x)\Big). \tag{4.14}$$

La figure 4.2 schématise l'architecture générale d'un tel modèle.

Stratégie basée sur la TFCT

Dans les modèles de masquages exposés plus haut, l'encodeur et le décodeur sont une généralisation de la TFCT. En effet, on peut voir la TFCT et son inverse comme des couches de convolutions prédéfinies comme on peut en discuter dans la section 3.2.2. Dans ce cas là, le séparateur aspirera à construire des masques comme ceux présentés en partie 4.2.1. Les premiers modèles d'apprentissage profond proposés pour résoudre la tâche de séparation de source utilisent des réseaux de neurones optimisés selon un critère de proximité entre les masques estimés et les masques idéaux [46, 47].
Ces premières approches se concentrent sur des masques d'amplitude et négligent la phase. Or, ignorer la phase ne permet pas de reconstruire convenablement le signal quand les différentes sources ont des amplitudes similaires. L'estimation de la phase est nécessaire pour obtenir une reconstruction plus précise. On pourrait essayer d'estimer le spectrogramme de phase à partir de l'estimation du spectrogramme d'amplitude en utilisant un algorithme de reconstruction de phase comme celui de Griffin-Lim [7]. Cependant, cet algorithme est très sensible aux erreurs contenues dans l'estimation de l'amplitude [50].
C'est pourquoi plusieurs travaux ont étendu l'estimation d'un masque d'amplitude vers des modèles estimant également la phase. Plusieurs types d'algorithmes ont ainsi été proposés :

- soit en utilisant des méthodes géométriques pour corriger la phase du mélange [51],
- soit en utilisant une branche dédiée à la reconstruction de la phase dans le réseau de neurones [52],
- soit en estimant directement un masque complexe [53].

Stratégie temporelle

On peut échapper aux problématiques d'estimation de la phase en utilisant des modèles dont les parties d'encodage f_e et de décodage f_s sont apprises.
Les modèles issus de ce type de stratégies sont appelés *Time-domain Audio Separation Network* (TasNet) issus de l'article du même nom [49]. Cette architecture, et plus particulièrement sa version convolutionelle [48] ont permis un gain de performance important dans la tâche de séparation de sources voix-voix.
Les encodeurs et décodeurs sont remplacés par des convolutions 1D et convolutions 1D inverses présentées dans la partie 3.2.2 :

$$f_e(x) = x \circledast W, \text{ avec } W \in \mathbb{R}^{1,C,L} \ , \tag{4.15}$$

$$f_d(X \cdot m_q) = (X \odot m_q) \circledast^{-1} V, \text{ avec } V \in \mathbb{R}^{C,L,1}, \tag{4.16}$$

où $\circledast^{-1}$ correspond à une convolution suivie d'une procédure d'*overlapp and add* 2.2.1.

Même si ces modèles sont appelées *time-domain*, ils résolvent bien la problématique de séparation dans un domaine de projection analogue à celui de la TFCT. C'est pourquoi, ce type de modèle est un objet d'étude privilégié pour comprendre le type de représentation utile au réseau de neurones de masquages.

La différence principale réside dans la projection apprise qui n'admet pas de structure ordonnée entre les différents canaux contrairement à la TFCT dont les noyaux sont indexés par le paramètre de fréquence. Dans la suite, on parlera de *front-end* lorsque l'on étudiera spécifiquement l'encodeur de ce type de réseau.

Type de séparateur

Pour les modèles de masquage, le choix de l'architecture du séparateur est un élément crucial pour la qualité de l'estimation des masques. On peut les classer en deux catégories :

— les séparateurs 1D,
— les séparateurs 2D.

Dans le premier cas, le séparateur ne prend en compte explicitement que la structure temporelle du signal. Ce type de séparateur peut prendre la forme d'un RNN [46, 49], d'un RNN "dual-path" [54], d'un CNN-1D dont les convolutions sont dilatées [48], ou d'une cascade de UNet [55]. Seule la TFCT permet l'utilisation de séparateurs 2D, car elle induit un ordre naturel sur les canaux fréquentiels.

4.3 Métriques de performances

Présentons maintenant quelques métriques utiles pour mesurer la qualité de reconstruction, et, pour certaines, utilisables également comme fonction de coût lors de l'optimisation d'un DNN.

4.3.1 Métriques spectrales

On appelle métriques spectrales, les métriques comparant des signaux de référence s_q à des signaux estimés $\hat{s}_q$ dans le domaine de la TFCT [56, 57].
Par exemple, la métrique spectrale complexe $\mathcal{L}_{\text{Complex}}$ est définie par :

$$\mathcal{L}_{\text{Complex}}(s_q, \hat{s}_q) = \frac{1}{FT} \sum_{n,k} \left|\left|\text{TFCT}_{s_q}[f,t] - \text{TFCT}_{\hat{s}_q}[f,t]\right|\right|^2. \tag{4.17}$$

Dans ce cas-ci, on considère l'information de phase pour mesurer la qualité de reconstruction.
Dans certains cas, il peut être intéressant de négliger la phase et de simplement se référer à la différence des spectrogrammes d'amplitude. Dans ce cas on définie la métrique $\mathcal{L}_{\text{Amp}}$ telle que :

$$\mathcal{L}_{\text{Amp}}(s_q, \hat{s}_q) = \frac{1}{FT} \sum_{f,t} \left|\left||\text{TFCT}_{s_q}[f,t]|^l - |\text{TFCT}_{\hat{s}_q}[f,t]|^l\right|\right|^2, \tag{4.18}$$

où l est un réel positif permettant de compresser l'échelle d'amplitude. On peut également prendre la fonction de coût sur le spectrogramme d'amplitude dans le domaine des

décibels :

$$\mathcal{L}_{\text{dB}}(s_q, \hat{s}_q) = \frac{1}{NK} \sum_{n,k} \left|\left|10\log_{10}(|\text{TFCT}_{s_q}[f,t]|) - 10\log_{10}(|\text{TFCT}_{\hat{s}_q}[f,t]|)\right|\right|^2. \quad (4.19)$$

Enfin, on peut prendre plusieurs niveaux de résolution [34] en prenant une famille de $(\text{TFCT}_l)_{l\in l_0,\dots,l_R}$ ayant des fenêtres d'analyses w_a de taille variable l_r :

$$\mathcal{L}_{\text{MultiRes}}(s_q, \hat{s}_q) = \frac{1}{R} \sum_{r=0}^{R-1} \mathcal{L}_{\square}^{\text{TFCT}_{l_r}}(s_q, \hat{s}_q). \quad (4.20)$$

où $\mathcal{L}_{\square}^{\text{TFCT}_{l_r}}$ correspond à une des fonctions de coût spectrales définies précédemment utilisant des fenêtres d'analyses de tailles l_r dans le calcul de leur TFCT.

4.3.2 Métriques temporelles

Pour définir des métriques de ressemblance directement dans le domaine temporel, Vincent et al. [58] propose de décomposer les distorsions d'un signal $\hat{s}_q$ selon plusieurs types :

— les distorsions dues aux autres sources, e_{interf},
— les distorsions dues à l'algorithme, e_{artif},
— les distorsions dues aux bruits, e_{noise}.

Ainsi, le signal $\hat{s}_q$ peut s'écrire sous la forme suivante :

$$\hat{s}_q(t) = \alpha s_q(t) + e_{\text{interf}}(t) + e_{\text{artif}}(t) + e_{\text{noise}}(t), \quad (4.21)$$

où le réel α correspond à un facteur de redimensionnement. Notons que l'on peut décomposer la distorsion e_{interf} en fonctions des autres sources $s_{q'}$ tel que :

$$e_{\text{interf}} = \sum_{q'\in Q\setminus q} e_{s'_q}(t). \quad (4.22)$$

Dans l'hypothèse que les différents termes des équations 4.21 et 4.22 sont orthogonaux, on peut les calculer de la façon suivante :

$$e_{s'_q}(t) = \frac{\langle \hat{s}_q | s'_q \rangle}{||s'_q||^2} s'_q(t), \quad (4.23)$$

$$e_{\text{noise}}(t) = \frac{\langle \hat{s}_q | n \rangle}{||n||^2} n(t), \quad (4.24)$$

$$\tilde{s}_q(t) = \alpha s_q(t) = \frac{\langle \hat{s}_q | s_q \rangle}{||s_q||^2} s_q(t), \quad (4.25)$$

$$e_{\text{artif}}(t) = \hat{s}_q(t) - (\tilde{s}_q(t) + e_{\text{interf}}(t) + e_{\text{noise}}(t)). \quad (4.26)$$

On peut finalement construire les métriques suivantes :

$$\text{Source to Distorsion Ratio :} \quad \text{SDR}(\hat{s}_q) = 10\log_{10}\frac{||\tilde{s}_q||^2}{||e_{\text{noise}} + e_{\text{artif}} + e_{\text{interf}}||^2}, \tag{4.27}$$

$$\text{Source to Interferance Ratio :} \quad \text{SIR}(\hat{s}_q) = 10\log_{10}\frac{||\tilde{s}_q||^2}{||e_{\text{interf}}||^2}, \tag{4.28}$$

$$\text{Source to Noise Ratio :} \quad \text{SNR}(\hat{s}_q) = 10\log_{10}\frac{||\tilde{s}_q + e_{\text{interf}}||^2}{||e_{\text{noise}}||^2}, \tag{4.29}$$

$$\text{Source to Artefact Ratio :} \quad \text{SAR}(\hat{s}_q) = 10\log_{10}\frac{||\tilde{s}_q + e_{\text{noise}} + e_{\text{interf}}||^2}{||e_{\text{artif}}||^2}. \tag{4.30}$$

Chacune de ces métriques permet de mesurer la quantité de distorsions relatives aux combinaisons linéaires présentes dans les dénominateurs des équations ci-dessus. Par exemple, le SDR mesure la quantité totale de distorsion présente dans le signal $\hat{s}$ et est régulièrement utilisé comme fonction de coût lors de l'optimisation de réseaux de neurones. On trouvera souvent la dénomination Si-SNR (pour *Scale-invariant Source-to-Noise Ratio*) comme synonyme du SDR (et non du SNR).
Notons que ces métriques ne permettent pas de prendre en compte l'invariance de perception liée à la phase. Par exemple, dans le cas extrême où $\hat{s}_q(t) = \mathcal{H}(s_q)(t)$, avec $\mathcal{H}$ la transformée de Hilbert (définie dans le chapitre 6), le SDR serait de $-\infty$ mais la perception humaine de s_q et de $\hat{s}_q$ serait identique.

4.3.3 Métriques perceptives

Afin de prendre en compte la perception que l'on se fait d'un signal, plusieurs métriques ont été spécialement conçues pour mesurer la qualité de restauration pour des signaux de voix. En particulier, le *Perceptual Evaluation of Speech Quality* [59] (ou PESQ donnant un score entre -.5 et 4.5) et le *Short-Time Objective Intelligibility* [60] (ou STOI donnant un score entre 0 et 1) mesurent respectivement la qualité et l'intelligibilité des signaux reconstruits.
On peut également utiliser l'échelle MOS (pour *Mean Opinion Score*) qui permet de caractériser la qualité de reconstruction d'un algorithme en sondant une population d'utilisateurs à donner des notes sur des échantillons de signaux. Ce type de métriques est par nature celle qui se rapproche le plus de l'expérience subjective que l'on se fait des signaux audio, mais est compliquée à mettre en place.

4.4 Bases de données

Dans la suite de cette thèse, les différentes expériences et optimisations seront effectuées sur des bases de données de séparation de sources classiquement utilisée dans la communauté. On détaille dans cette section la base de données LibriMix [61] et la base de données du Deep Noise Suppresion Challenge [62] (DNS).

4.4.1 Base de données LibriMix

Cette base de données de séparation de sources voix-voix est constituée de superpositions de signaux parlés issus de la base de données LibriSpeech [63]. Cette base de données contient 921 voix de personnes différentes dans la base d'entraînement. Des bases de données plus larges, comme VoxCeleb [64] contiennent davantage de voix, mais ne garantissent pas l'absence de bruits parasites. Ainsi, des biais potentiellement indésirables peuvent être appris avec l'utilisation de ce type de bases. C'est pourquoi on fait généralement appel à LibriMix pour optimiser et évaluer les modèles de séparation de sources. Même si l'on pourrait enrichir les bases de données de séparation de sources à partir d'autres bases de voix afin d'obtenir des modèles plus performants et généralisant mieux, la simplicité de cette base de données en fait un choix idéal.
Dans les expériences de séparation de sources définies dans les chapitres suivants, les mélanges issus LibriMix seront construits dynamiquement lors des apprentissages. De cette manière, on augmente radicalement le nombre d'exemples possibles par rapport à la base de données d'origine. L'ensemble des exemples utilisés dans les phases d'apprentissage, de validation ou de test seront tirés afin d'assurer une superposition simultanée de deux voix provenant de locuteurs différents.

4.4.2 Base de données DNS

La base de données du Deep Noise Suppresion challenge (DNS) est conçue pour des tâches de rehaussement de la parole. Cette base de données est constituée de trois sous-ensembles.
Le premier, le sous-ensemble de bruit, est constitué essentiellement de bruit à partir des bases de données AudioSet [65] et FreeSound [1]. Il s'agit de bruit ou d'évènement sonores issus de 150 classes distinctes de la voix humaine.
Le deuxième, le sous-ensemble de voix, est composé de voix issues de plusieurs bases de données. Les différents locuteurs sont sélectionnés pour assurer une diversité importante dans les langues représentées ainsi que dans les contextes d'enregistrement.
Le dernier, le sous-ensemble de réponses impulsionnelles, permet d'augmenter la diversité de la base de données en modifiant les mélanges de voix et bruits par l'application d'une convolution entre le mélange et une réponse impulsionnelle donnée correspondant à celle d'une pièce ou d'un environnement donné.
Cette base de données est largement utilisée pour comparer les performances des modèles sur la tâche de rehaussement de la parole car elle est issue du challenge annuel de rehaussement de la parole des plus grandes conférences internationales de traitement de la parole (InterSpeech jusqu'à 2021 et ICASSP depuis).

1. https ://freesound.org

4.5 Conclusion

Les modèles d'apprentissage profond constituent aujourd'hui l'état de l'art pour résoudre les tâches de séparation de sources et de rehaussement de la parole. La méthode la plus couramment utilisée et étudiée consiste en la construction d'un masque à appliquer à une représentation 2D du signal comme la TFCT. L'utilisation de la TFCT entraîne la nécessité d'analyser et de synthétiser la phase pour obtenir un niveau de séparation maximal. Pour se faire, on peut d'abord essayer de modéliser la phase de la TFCT. On peut également essayer de s'abstraire de la représentation complexe en utilisant des bancs de filtres réels afin de n'apprendre qu'un masque réel. On peut également essayer d'utiliser des méthodes temporelles afin de modifier le signal directement dans ce domaine sans utiliser de représentation temps-fréquences (ou assimilées) nécessitant un paramètre de phase pour localiser les différentes composantes du signal. On a également vu que le choix de la fonction de coût est aussi un paramètre important afin d'optimiser les modèles d'apprentissage profond. L'utilisation d'une fonction possédant ou non une invariance à la phase peut amener à des performances très différentes en fonction de la structure des modèles étudiées.
Enfin, on a présenté les bases de données que l'on utilisera dans les chapitres suivants pour mener nos expériences.

Deuxième partie

Contributions

Chapitre 5

Analyse des réseaux de neurones à valeurs complexes

5.1 Introduction

Comme on l'a vu dans les parties précédentes, la TFCT d'un signal appartient au domaine complexe, il paraît naturel de tenter de la traiter dans un réseau de neurones à valeurs complexes afin d'utiliser cette structure au fil des différentes transformations élémentaires effectuées par un modèle d'apprentissage profond. L'objectif de ce chapitre est de présenter deux modèles complexes pour résoudre les tâches de séparation de sources et de rehaussement de la parole.

5.1.1 État de l'art

Plusieurs modèles ont été proposés dans le domaine complexe pour résoudre la tâche de séparation de sources dans le domaine complexe. Les modèles Deep-Complex UNet [66] et Deep Complex Convolution Recurrent Network (DCCRN) sont deux exemples de modèles à valeurs complexes appliqués à la tâche de séparation de sources (ce dernier ayant même obtenu les meilleures performances dans le challenge DNS de 2020 [67]). Ces deux modèles reposent sur une stratégie de masquage décrite génériquement dans la partie 4.2.2. L'intérêt principal de ces modèles est d'estimer directement des masques complexes à appliquer à la TFCT du mélange, permettant potentiellement de reconstruire parfaitement la TFCT des signaux cibles comme on a pu le décrire précédemment dans la partie 4.2.1.

UNet Complexe

L'architecture UNet complexe dans un cadre de rehaussement de la parole a été proposée par Choi et al. [66].
Ce modèle prend en entrée la TFCT complexe du signal à améliorer. Le UNet est composé de couches convolutionnelles 2D appliquées sur les différentes représentations latentes avec un pas d'avancement de 1 sur la dimension temporelle et un pas d'avancement de 2 sur la dimension fréquentielle.

Étant donnée la représentation latente $X_l \in \mathbb{C}^{C_l,F_l,T}$ de la couche l, où C_l représente le nombre de canaux, F_l la dimension fréquentielle et T la dimension temporelle, la couche de convolution $l+1$ est définie par :

$$X_{l+1} = \mathbb{C}\text{PReLU}\Big(\mathbb{C}\text{-BN}\big(X_l \circledast_{\mathbb{C}} W_{l+1}\big)\Big), \tag{5.1}$$

où $\mathbb{C}$PReLU correspond à la version $\mathbb{C}$ReLU de la non-linéarité PReLU [68], $\mathbb{C}$-BN à la batch normalisation complexe et $W_{l+1} \in \mathbb{C}^{C_l,C_{l+1},3,2}$ le tenseur de convolution de la couche $l+1$. Finalement, la représentation X_{l+1} est un tenseur complexe de dimension $C_{l+1} \times F_l/2 \times T$.
La partie décodeur du modèle est le symétrique de cet encodeur où l'opération de convolution complexe est remplacée par une opération de convolution transposée complexe.

DCCRN

Le modèle *Deep Complex Convolutional Reccurent Neural Networks* (DCCRN) constitue l'état de l'art dans le domaine de rehaussement de la parole [67]. Il s'agit d'un UNet comme décrit plus haut muni d'un LSTM complexe en tant que *bottleneck* entre l'encodeur et le décodeur. Le Split-LSTM utilisé dans ce modèle correspond à la version LSTM du Split-RNN complexe décrit dans l'équation 3.22 tel que pour une donnée $X \in \mathbb{C}^{C,T}$ et un RNN complexe $\text{LSTM}_{\mathbb{C}}$ on ait :

$$\text{LSTM}_{\mathbb{C}}(X) = \Big(\text{LSTM}_{\Re}(\Re(X)) - \text{LSTM}_{\Im}(\Im(X))\Big) + i\Big(\text{LSTM}_{\Im}(\Re(X)) + \text{LSTM}_{\Re}(\Im(X))\Big), \tag{5.2}$$

où $\text{LSTM}_{\Im}$ et $\text{LSTM}_{\Re}$ sont des LSTM réels. Le bottleneck ne traite donc pas la représentation X dans le domaine complexe. En effet, les différentes représentations cachées des LSTM considèrent les parties réelle et imaginaire indépendamment.

5.1.2 Problématique

Ce dernier modèle à valeurs complexes permet d'obtenir de très bonnes performances pour la tâche de rehaussement de la parole. Cependant, en regardant en détail son architecture, il n'est pas clair qu'elle exploite les propriétés algébriques de $\mathbb{C}$. Nous allons donc chercher ici à comparer cette architecture à des équivalents en nombres réels pour valider ou non l'intérêt de l'utilisation de nombres complexes dans les DNN.

5.2 Validation des UNet complexes

L'objectif de cette section est de comparer le modèle DCCRN à des équivalents réels en analysant la représentation de la phase dans les modèles complexes et proposant une architecture réelle adaptée à la jointure des parties réelles et imaginaires dans le modèle.

5.2.1 Représentation de la phase dans DCCRN

Intéressons-nous ici à la représentation des données dans le modèle DCCRN. Notamment en nous demandant comment l'information de phase peut être encodée dans les différentes couches du réseau.

Phase et ReLU

La non-linéarité $\mathbb{C}$ReLU traite les parties réelle et imaginaire d'un nombre complexe quelconque de manières indépendantes. Ainsi, si on considère un nombre complexe z, la phase de celui-ci influera sur l'amplitude après application de la $\mathbb{C}$ReLU, c'est-à-dire qu'il existe des angles ψ tels que :

$$|\mathbb{C}\text{ReLU}(z)| \neq |\mathbb{C}\text{ReLU}(ze^{i\psi})|. \tag{5.3}$$

En revanche, l'angle est invariant par application d'un facteur de dilatation. C'est-à-dire qu'on a :

$$\arg(\mathbb{C}\text{ReLU}(z)) = \arg(\mathbb{C}\text{ReLU}(\alpha z)), \quad \alpha \in \mathbb{R}^{+*}, \tag{5.4}$$

Ainsi, on constate que l'amplitude varie avec la phase de l'entrée alors que des variations d'amplitude ne peuvent pas permettre d'apporter des modifications dans la phase. Les couches linéaires sont quant à elles invariantes à la phase. On peut donc se demander légitimement comment la phase peut être traitée de manière naturelle à travers la structure complexe dans un tel modèle.

TFCT

A partir de la TFCT d'un mélange x, on peut comparer les représentations latentes induites par $X_0 = \text{TFCT}(x)$ et $X_0^\psi = X_0 e^{i\psi}$. Les masques idéaux complexes définis en parties 4.2.1 à appliquer à X_0 et à X_0^ψ sont strictement identiques.
En comparant les statistiques des différentes représentations latentes X_l et X_{l+1} sur toutes les couches du modèle, on peut voir dans la figure 5.1 que le modèle tend effectivement à obtenir les mêmes masques. Cependant, le modèle réussit à obtenir cette identité à la seule et dernière couche, on peut donc se demander si c'est bien la structure complexe du modèle qui permet d'obtenir de telles performances. On peut noter que l'amplitude des représentations latentes convergent vers les mêmes valeurs jusqu'à l'avant-dernière couche. Néanmoins, comme des variations importantes induites par le résidu de la première couche sont appliqués dans la dernière couche, l'écart type réaugmente fortement et les masques des deux entrées diffèrent.
Notre hypothèse est que c'est le fait de traiter les parties réelle et imaginaire sur une dimension dédiée qui permet au modèle d'être aussi expressif. Pour valider cela, nous définissons ici une architecture analogue à l'architecture complexe mais prenant en compte les valeurs complexes comme des vecteurs de $\mathbb{R}^2$. Nous appelons cette architecture un réseau de neurones de $\mathbb{R}^2$.

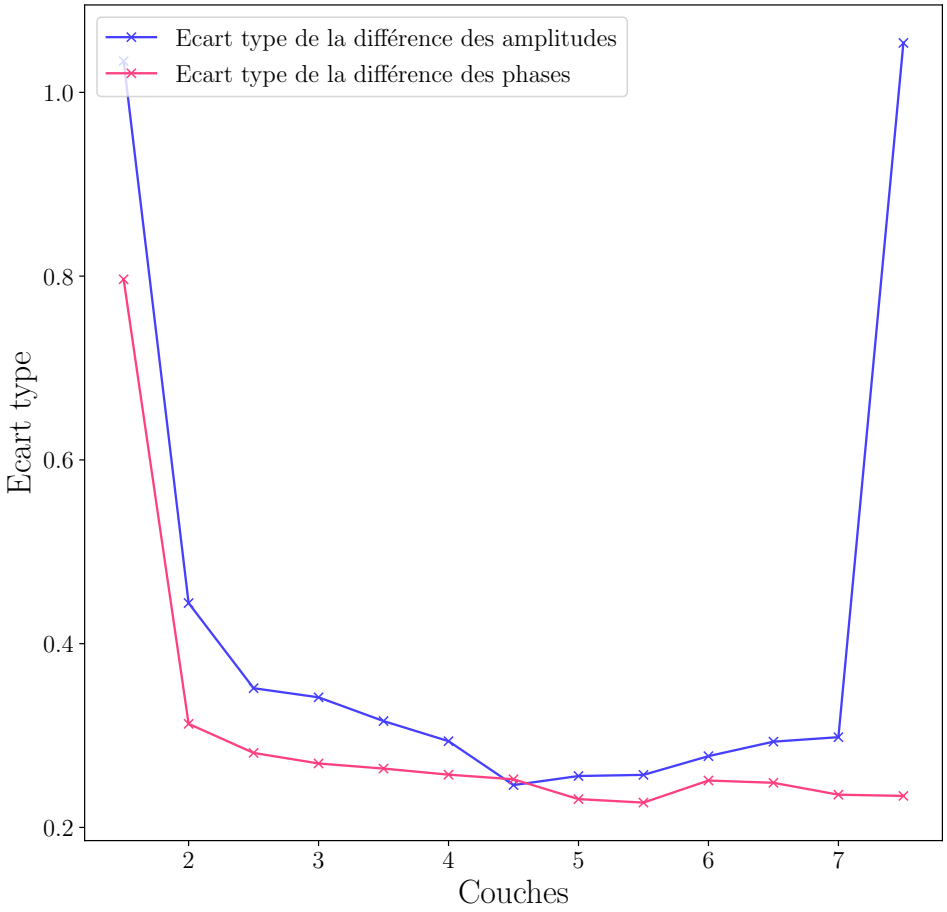


FIGURE 5.1 – Ecart type de la différence des états intermediaires pour deux entrées X_0 et $X_0^{\psi=\pi/2}$. Comme on peut le voir, l'amplitude de l'estimation final change fortement en fonction de la phase absolue $\psi = \pi/2$ de l'entrée.

5.2.2 Réseau de neurones dans $\mathbb{R}^2$

Dans cette partie, on tente de remplacer les transformations complexes de DCCRN par des transformations plus générales de $\mathbb{R}^2$ dans $\mathbb{R}^2$.
On peut assimiler un nombre complexe z à un vecteur de $\mathbb{R}^2$ et une fonction linéaire complexe à une matrice 2×2 de la forme :

$$\begin{pmatrix} a & b \\ -b & a \end{pmatrix} \tag{5.5}$$

Dans le cadre de notre modèle, on remplace l'ensemble des poids complexes par des poids de $\mathbb{R}^{2\times 2}$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}. \tag{5.6}$$

Convolutions de $\mathbb{R}^2$

De la même façon, on remplace les convolutions complexes par des convolutions de $\mathbb{R}^2$ définies par une matrice de poids W composé de 4 matrices de poids $W_{a,b,c,d} \in \mathbb{R}^{C_1,C_2,K}$:

$$W = \begin{pmatrix} W_a & W_b \\ W_c & W_d \end{pmatrix}, \tag{5.7}$$

et en se munissant d'un vecteur $x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$, avec $x_0, x_1 \in \mathbb{R}^{C_1,T}$ on définit la convolution de $\mathbb{R}^2 \to \mathbb{R}^2$ par :

$$x \circledast_{\mathbb{R}^2} W = \begin{pmatrix} x_0 \circledast W_a + x_1 \circledast W_c \\ x_0 \circledast W_b + x_1 \circledast W_d \end{pmatrix}. \tag{5.8}$$

Cette définition de la convolution entraîne un nombre de FLOPS équivalent à la convolution complexe, mais aboutit à un nombre de paramètres deux fois plus grands.

LSTM de $\mathbb{R}^2$

De façon encore similaire, on remplace le LSTM complexe présentée dans la section précédente par un $\text{LSTM}_{\mathbb{R}^2}$ composé de quatre LSTM réels :

$$\text{LSTM}_{\mathbb{R}^2} = \begin{pmatrix} \text{LSTM}_a & \text{LSTM}_b \\ \text{LSTM}_c & \text{LSTM}_d \end{pmatrix}, \tag{5.9}$$

l'application à un vecteur x de $\mathbb{R}^2$ donne :

$$\text{LSTM}_{\mathbb{R}^2}(x) = \begin{pmatrix} \text{LSTM}_a(x_0) + \text{LSTM}_c(x_1) \\ \text{LSTM}_b(x_0) + \text{LSTM}_d(x_1) \end{pmatrix}. \tag{5.10}$$

De la même manière que pour les convolutions, le nombre de FLOPS est identique au cas complexe, mais le nombre de paramètres est multiplié par 2.

Non-linéarité et Batch-Norm

Les non-linéarités et batch-normalisation sont gardées telles quelles :
— la PReLU est appliquée terme à terme sur les deux dimensions,
— la batch-normalisation complexe correspond à un blanchiment d'un ensemble de vecteurs de $\mathbb{R}^2$, on peut donc directement l'appliquer dans notre modèle.

On appelle ce modèle DR2CRN pour *Deep $\mathbb{R}^2$ Convolutional Reccurent Network*.

5.2.3 Résultats expérimentaux

Dataset

Nous utilisons l'ensemble de données du challenge DNS [67] présenté dans la partie 4.4. Les exemples sont créés à la volée à partir de différentes voix, d'arrière-plans

et de réponses impulsionnelles avec un SNR compris entre -5 et +20 dB à 16kHz.

Caractéristiques des modèles

On cherche à comparer un modèle DCCRN et un modèle DR2CRN. Les deux modèles ont exactement le même nombre de couches (6) et le même nombre de canaux à chacune de celles-ci (table 5.1). L'entrée des modèles est une TFCT calculée sur des fenêtres de taille 400 avec un pas d'avancement de taille 100. Afin de s'assurer que c'est bien cette structure d'appariement des parties réelle et imaginaire qui bénéficie à ce type de modèle, on compare également ces deux modèles à un modèle purement réel utilisant les informations des parties réelle et imaginaire dans des canaux indépendants. On appellera ce modèle Deep Convolution Recurent Network (DCRN). Le nombre de FLOPS est constant pour les trois modèles testés.

TABLE 5.1 – Spécification du nombre de canaux dans les modèles DCCRN, DR2CRN et DCRN.

Couches	1	2	3	4	5	6
# Canaux	16	32	64	128	128	128

Protocole expérimental

Les entraînements des différents modèles sont réalisés avec des mélanges de 3 secondes. La fonction de coût Si-SNR est utilisée pour optimiser les modèles avec un taux d'apprentissage de 10^{-3} à l'aide de l'optimiseur Adam [69].

Performances

Les performances des deux modèles sont présentées dans la table 5.2. On peut voir que les modèles complexe (DCCRN) et réel (DCRN) performent de manière équivalente. De plus, le modèle à valeurs dans $\mathbb{R}^2$ permet d'obtenir des performances légèrement mais significativement plus élevées que les deux autres en termes de Si-SNR de STOI et de PESQ. Ainsi, nous avons établi expérimentalement que ce n'est pas la structure algébrique complexe qui confère à DCCRN son pouvoir de représentation mais le fait que la partie réelle et la partie imaginaire sont traitées à chaque étape comme deux composantes liées plutôt que comme deux canaux indépendants.

Conclusion

L'hypothèse selon laquelle la structure complexe ne semble pas réellement apporter quelque chose pour l'apprentissage profond semble donc vérifiée avec ce type d'architecture complexe utilisant des non-linéarités influant sur les réponses complexes de manière différente selon l'amplitude et la phase. Cependant, on peut se demander si une architecture traitant la phase et l'amplitude de manière plus stable obtiendrait de meilleures performances.

TABLE 5.2 – Performances des modèles DCCRN, DR2CRN accompagné de leur écart type sur la base de données d'évaluation. L'écart type est révélateur des variations de performances en fonction du rapport signal sur bruit plus que de l'incertitude sur l'estimation des performances moyennes. Les modèles sont paramétrés de telle façon que le nombre de FLOPS soit constant pour tous les modèles. Il apparaît que le modèle DR2CRN performe légèrement, mais significativement mieux que ses contreparties complexes et réelles.

Modèle	SiSNR	STOI	PESQ
DCCRN	15.56 ± 2.69	0.89 ± 0.11	3.16 ± 0.58
DR2CRN	16.08 ± 2.77	0.90 ± 0.10	3.21 ± 0.60
DRCRN	15.67 ± 2.71	0.90 ± 0.10	3.17 ± 0.60

5.3 TasNet à valeurs complexes

Essayons donc maintenant de redéfinir avec un séparateur à valeurs complexes l'architecture TasNet [49] présentée dans la partie 4.2.2.
Dans le cas réel, l'encodeur et le décodeur du modèle TasNet sont des convolutions 1D permettant de projeter le signal sur un banc de filtres réels appris. Dans ces conditions, le séparateur, une succession de LSTM, a pour objectif de construire un masque réel à appliquer sur le signal encodé avant l'étape de décodage.
L'objectif de cette section est de construire un modèle complexe suivant l'architecture TasNet permettant d'obtenir une invariance à la phase.

5.3.1 RNN complexe

Comme on a pu en discuter dans la section précédente, les modèles complexes utilisant des $\mathbb{C}$ReLU ne permettent pas de représenter la phase convenablement. On cherche ici à construire un type de séparateur utilisant des non-linéarités invariantes à la phase.
De plus, les RNNs complexes présentés jusqu'ici n'étaient pas complexes à proprement parler. On présente ici des méthodes reposant sur la construction de RNN dont les matrices de poids associées sont à valeurs complexes.

RNN unitaire

Une matrice $W \in \mathbb{R}^{C\times C}$ est dite orthogonale si $WW^T = \mathbf{Id}$. Cette notion s'étend dans le domaine complexe : une matrice $W \in \mathbb{C}^{C\times C}$ est dite unitaire si elle vérifie l'égalité $WW^* = \mathbf{Id}$ avec W^* la matrice transposée conjuguée de W.
Les matrices unitaires permettent de conserver la norme, c'est-à-dire que pour un vecteur $x \in \mathbb{C}^C$ on a :

$$\|Wx\| = \|x\|. \tag{5.11}$$

Dans le cas réel, il a été montré qu'initialiser des matrices de poids d'un RNN sur des matrices orthogonales permet d'obtenir des modèles plus stables lors de l'apprentissage [70]. Dans l'article Unitary Evolution Recurrent Neural Network [19] Arjovksy et al. montrent que si l'équation d'évolution interne d'un RNN vérifie :

$$h_{t+1} = \mathrm{ReLU}(Vx_{t+1} + Wh_t), \tag{5.12}$$

avec la matrice de poids $W \in \mathbb{C}^{C\times C}$ unitaire, alors la norme du gradient après rétropropagation est bornée par :

$$\left\|\frac{\partial \mathcal{L}}{\partial h_t}\right\| \leq \left\|\frac{\partial \mathcal{L}}{\partial h_T}\right\| \prod_{k=t}^{T-1} \|D_{k+1}\| \leq \left\|\frac{\partial \mathcal{L}}{\partial h_T}\right\|, \tag{5.13}$$

où D_{k+1} correspond à la matrice associée à la Jacobienne de la non-linéarité ReLU et où $\|.\|$ correspond au rayon spectral pour D_{k+1} et à la norme L_2 pour le reste.
Ainsi, en contraignant l'apprentissage de W dans le domaine des matrices unitaires, on assure que le gradient ne peut pas exploser durant l'apprentissage. Les auteurs de [19] étendent ce résultat en utilisant une non-linéarité modReLU définie par :

$$\mathrm{modReLU}(z|b) = \mathrm{ReLU}(|z| - b)e^{i\arg z}. \tag{5.14}$$

Un autre intérêt de cette non-linéarité est qu'elle est invariante à la phase, une propriété que l'on cherche à obtenir dans notre problématique de séparation de sources.

Extension des RNN unitaires

Cependant, les RNN unitaires définis dans l'article de Arjovsky et al. [19] sont restreints à une sous-variété des matrices unitaires.
Wisdom et al. [71] généralisent leur concept à l'ensemble des matrices unitaires en utilisant une optimisation agissant directement sur cet espace. Pour plus de détails sur cette méthode, on peut se référer aux notes de Hemant D. Tagare [72].
A partir de cette stratégie d'optimisation, l'architecture cgRNN (pour *Complex Gated Recurrent Neural Networks*) proposée par Wolter et al. [73] étend le principe des RNN unitaires aux GRU.
Ces briques vont nous permettre de définir un modèle complexe ayant de bonnes propriétés :

- de représentation long terme à travers l'utilisation de matrice unitaire.
- de représentation de la phase à travers l'utilisation de non-linéarité modReLU.

5.3.2 TasNet complexe

On se propose donc de construire des TasNet à valeurs complexes en utilisant plusieurs paradigmes de RNN complexes.

Encodeur et décodeur complexes

L'encodeur et le décodeur des modèles complexes sont de simples convolutions 1D complexes comme définies dans la partie 3.3.2. La différence avec le modèle TasNet standard réside dans la suppression de la non-linéarité ReLU en sortie de l'encodeur, car on veut obtenir un modèle invariant à la phase.

RNN à poids complexes

Pour les RNN à poids complexes, on utilise ici des cgRNN. On testera d'une part, des modèles optimisés sur l'espace des matrices unitaires et d'autre part, des modèles optimisés avec une descente de gradient standard, mais initialisée sur l'espace des matrices unitaires. Les cgRNN utilisés ici sont définis de la même manière que dans l'article [73].

Split-GRU

On compare également ces RNN complexes à des split-GRU l'équivalent en GRU des Split-LSTM présentées dans la section précédente :

$$\mathrm{GRU}_{\mathbb{C}}(X) = \Big(\mathrm{GRU}_{\Re}(\Re(X)) - \mathrm{GRU}_{\Im}(\Im(X))\Big) + i\Big(\mathrm{GRU}_{\Im}(\Re(X)) + \mathrm{GRU}_{\Re}(\Im(X))\Big), \tag{5.15}$$

où $\mathrm{GRU}_{\Im}$ et $\mathrm{GRU}_{\Re}$ sont des GRU réels.

5.3.3 Résultats expérimentaux

Dataset

Nous utilisons ici le dataset de séparation de sources LibriMix [61] présenté dans la partie 4.4. Les exemples sont créés à la volée à partir des différentes voix avec des SNR compris entre 0 et 5 dB pour la voix dominante. La fréquence d'échantillonnage des mélanges est fixée à 16kHz.

Caractéristiques des modèles

On compare ici des TasNet réels et complexes. Pour l'encodeur de l'ensemble des modèles, on utilise 512 filtres complexes (1024 pour le modèle réel) de tailles 256. Le nombre de FLOPS est équivalent pour l'ensemble des différents modèles. On compare dans cette section trois séparateurs différents :

- un séparateur GRU réel,
- un séparateur complexe du type Split-GRU,
- un séparateur complexe du type cgRNN optimisé selon deux méthodes (standard et unitaire).

TABLE 5.3 – Performances des différents TasNet complexes et réels. L'écriture $x \pm y$ correspond à une valeur moyenne x et un écart type y sur le jeu de données. L'écart type y s'explique essentiellement par la dispersion des valeurs de SNR dans l'ensemble de données de test.

Séparateur	SiSNR	STOI	PESQ
GRU réel	10.09 ± 3.55	2.63 ± 0.32	0.75 ± 0.07
Split-GRU	8.91 ± 3.52	2.64 ± 0.29	0.74 ± 0.07
cgRNN (optim standard)	8.45 ± 3.32	2.56 ± 0.31	0.73 ± 0.07
cgRNN (optim unitaire)	6.07 ± 3.32	2.45 ± 0.31	0.69 ± 0.08

Protocole expérimental

L'entraînement des modèles est réalisé avec des mélanges de 3 secondes. La fonction de coût Si-SNR est utilisée pour optimiser les modèles avec un taux d'apprentissage de 10^{-3} à l'aide de l'optimiseur Adam.

Performances

La table 5.3 présente les performances des différents modèles présentées ici. Comme on peut le voir, le modèle réel est celui qui performe le mieux sur toutes les métriques. Il est également intéressant de noter que parmi les modèles complexes, le modèle Split-GRU est celui qui performe le mieux, soit le modèle complexe admettant une structure éloignée d'une véritable structure complexe. Il semble donc que les structures complexes contraintes par les RNNs à valeurs complexes ne profitent pas au modèle TasNet. De plus, notons que plus l'on contraint la structure complexe dans les RNN, plus les performances du modèle chutent.

5.4 Conclusion

Dans la première partie de ce chapitre, on a comparé un modèle complexe efficient à un équivalent réel de celui-ci. On a pu donner des éléments de réflexions sur l'importance de l'algèbre complexe et les performances de ce dernier. Dans la seconde partie, on a tenté de construire un modèle complexe à partir d'un modèle réel. L'utilisation de RNN complexe bien défini ne permet pas d'améliorer les performances malgré les bonnes propriétés théoriques (unitaire, invariant à la phase) de ces modèles.
La modélisation de la phase de la TFCT ou d'une quelconque structure complexe ne semble pas permettre d'améliorer les performances pour ces tâches de séparation de sources et de rehaussement de la parole. Il faut donc trouver d'autres méthodes permettant de modéliser la phase dans les réseaux de neurones.

Chapitre 6

Apprentissage de bancs de filtres experts

Dans cette partie, on va s'intéresser aux réseaux de neurones utilisant une méthode de masquage pour résoudre la tâche de séparation de sources telle qu'elle a été présentée dans la section 4.2.2. Le modèle doit déterminer un masque à appliquer à une projection en deux dimensions du signal (comme la TFCT). Plus particulièrement, on s'intéresse à la construction d'un encodeur augmentant les bénéfices de l'utilisation des méthodes de masquage.

6.1 Introduction

Comme on a pu le décrire dans la section 4.2.2, on peut unifier les méthodes spectrales et les méthodes temporelles en remarquant que la TFCT est obtenue par un ensemble de paramètres particuliers et constants dans la phase d'apprentissage d'une convolution 1D. En traitement du signal, on a l'habitude d'appeler un tel ensemble de convolutions un *banc de filtres*. En toute généralité, on parlera de *frontends* audio pour désigner cette première étape de convolutions permettant la projection d'un signal à une dimension vers une représentation à deux dimensions. Lewicki et al. [74] montre que le choix du banc de filtres est important et dépend du type de sources étudiées. Kavalerov et al. [75] ont aussi montré que la TFCT permet d'obtenir des performances supérieures à l'apprentissage de filtres libres réels dans le cas de la séparation de sources universelles (séparation de deux sources quelconques). Il paraît donc pertinent de réfléchir à la façon de les paramétrer afin de maximiser les performances des modèles sur une tâche particulière.

6.1.1 Objectifs

L'objectif de ce chapitre est de déterminer les propriétés d'intérêt pour la tâche de séparation de sources et de comparer les différents front-ends proposés jusqu'ici dans leurs structures et leurs hyperparamétrisations (tailles des filtres, nombre de filtres...). On cherchera ensuite à construire des bancs de filtres paramétrés respectant ces propriétés. Enfin, on comparera les différents bancs de filtres étudiés dans ce chapitre sur

deux tâches de séparation de sources : la séparation de sources voix-voix et la séparation de sources voix-bruit.

6.1.2 Taxonomies

On distingue trois types de front-ends :

— Les front-ends fixes : dont les filtres sont fixés et connus à l'avance et n'évoluent pas dans la phase d'apprentissage [76].

— Les front-ends libres : dont les filtres peuvent prendre n'importe quelle forme et sont optimisés dans la phase d'apprentissage [49].

— Les front-ends contraints : dont les filtres sont paramétrés par un ensemble de contraintes permettant de construire des familles de formes d'ondes bien définies [77, 78].

Dans le dernier cas, les contraintes peuvent être de différentes natures :

— Des contraintes fortes, où les paramètres appris permettent de construire directement les filtres (fréquences de coupures, largeur de bande, ...).

— Ou des contraintes faibles [79], dans le cas où les filtres sont appris librement et sont ensuite modifiés via des transformations bien choisies.

Contraindre les filtres à avoir certaines formes permet d'assurer l'obtention de propriétés d'intérêt dans le banc de filtres appris, tout en permettant d'obtenir une certaine flexibilité sur la diversité des filtres obtenus que la TFCT ne peut pas obtenir.

6.1.3 Front-ends pour masques d'amplitude

L'idée d'apprendre des bancs de filtres (réels ou complexes) pour généraliser la TFCT de façon optimale pour une tâche donnée, apparait bien avant les méthodes temporelles décrites dans la section 4.2.2 pour la séparation de sources. Par exemmple, en 2011, Jaitly et al. [80] utilise une machine de Boltzmann restreinte pour apprendre une représentation de la voix.

Depuis, de nombreuses méthodes ont proposé d'utiliser des stratégies bout-en-bout (c'est à dire sans utilisation de transformations prédéfinies lors du traitement de la donnée) pour résoudre des tâches de classification comme la reconnaissance automatique de parole (ASR pour Automatic Speech Recognition) ou la reconnaissance de phonèmes [81, 82, 83, 84, 85] à l'aide de front-ends différents de la TFCT.

Plus récemment, des approches ont été développées afin d'initialiser les filtres sur des formes d'ondes connues comme les filtres de Gabor [78] utilisés dans le cadre de l'ASR ou des filtres Gammatones pour la tâche de séparation de sources [76] ce qui permet d'obtenir des performances supérieures aux approches sur spectrogrammes.

Venkataramani et al. [86] sont les premiers à proposer l'utilisation d'un banc de filtres appris pour la séparation de sources par des réseaux de neurones. Ils utilisent un banc de filtres réel constitué de cosinus discrets avec un pas d'avancement de 1 suivi d'une fonction de lissage pour estimer un masque d'amplitude. Ainsi, ils obtiennent un front-end apprenable adapté à la stratégie de masquage.

6.1.4 Frontends pour masques d'amplitude-phase

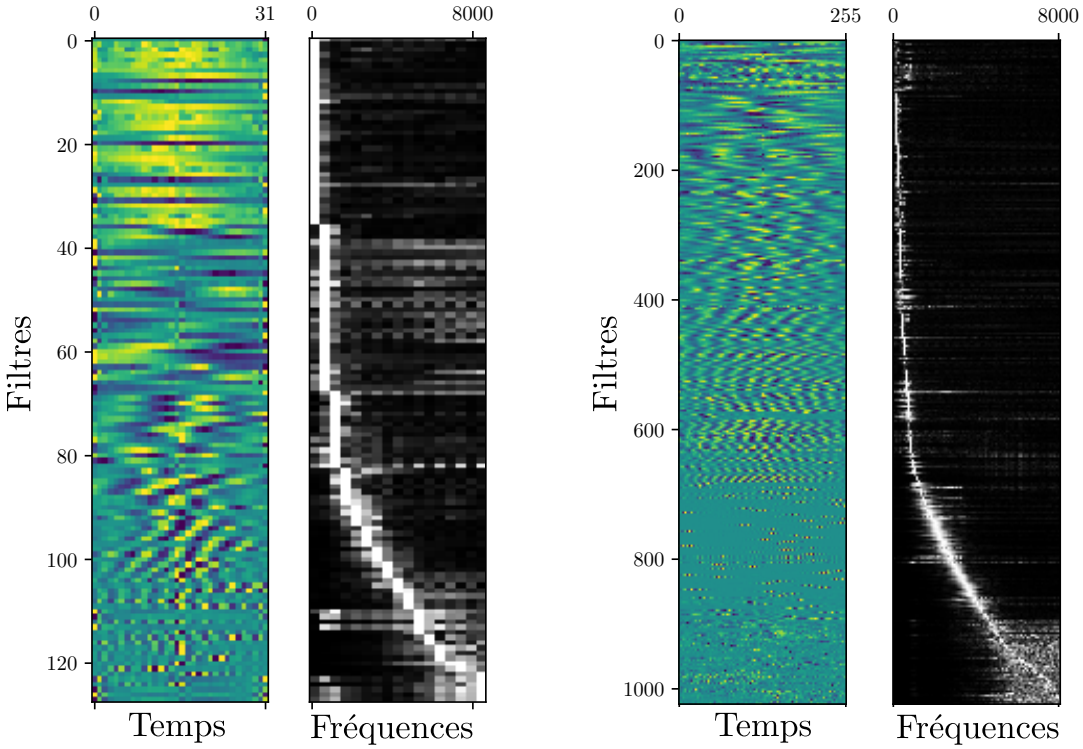
Afin d'estimer la phase et de l'intégrer dans la stratégie de masquage réel, Luo et al. [49] proposent d'utiliser une convolution 1D suivie d'une non-linéarité ReLU. La phase n'est pas représentée explicitement mais implicitement à travers les réponses de filtres différents ayant une réponse impulsionnelle proche. La figure 6.1 présente des filtres appris par un réseau de type Conv-TasNet [48] pour des tailles de filtres $L = 32$ et $L = 256$.
Dans cet article, les auteurs montrent qu'un séparateur réel a la capacité de fournir un masque réel permettant la reconstruction parfaite du signal s'il y a suffisamment de filtres pour représenter les différents décalages de phases. Cette représentation du signal constitue l'état de l'art pour la séparation de sources. Depuis, les principales avancées utilisant cette approche ont été faites sur l'architecture du séparateur comme le TCN [48, 87] (Temporal Convolutional Network), dual path RNN [54] ou plus récemment avec des transformers [88, 89].

6.1.5 Limitation

Ce type de frontend demande une grande quantité de filtres de petite taille. Les modèles les plus performants utilisent des filtres de convolutions de 16 échantillons (2 ms de signal) avec un nombre de filtres supérieur à 256. L'utilisation de 16 filtres devrait suffire pour reconstruire parfaitement le signal, mais le séparateur semble solliciter de l'information très redondante. Le nombre de filtres augmente la complexité du modèle au niveau de l'encodeur/decodeur mais augmente aussi indirectement la complexité du séparateur puisque qu'il faudra augmenter le nombre de canaux pour profiter de cette sur-paramétrisation.
L'utilisation de fenêtres de quelques échantillons ne permet pas d'effectuer une réduction importante de la dimension temporelle du signal. En effet, plus la taille de la fenêtre est petite, plus l'est également le pas d'avancement et plus la séquence en sortie de l'encodeur est longue. La complexité du séparateur augmente donc quand la taille des fenêtres diminue, et les performances de certaines architectures (comme les RNNs) peuvent aussi diminuer.
C'est pourquoi on aimerait construire des encodeurs permettant de diminuer le nombre de filtres dans le cas des petites fenêtres ou alternativement, d'augmenter la taille des fenêtres afin de réduire la complexité du séparateur.

6.2 Propriétés désirables

Dans cette partie, on s'intéresse aux propriétés intéressantes à contraindre dans l'apprentissage des bancs de filtres. En particulier, pour le cas de la séparation de sources, on identifie les propriétés émergeant naturellement lors de l'apprentissage de bancs de filtres dans le modèle Conv-TasNet. Ainsi, on peut construire des filtres contraints afin de faciliter l'apprentissage du modèle tout en obtenant de meilleures performances.



(A) Taille des filtres : 32 (B) Taille des filtres : 256

FIGURE 6.1 – Exemples de filtres appris pour filtres réels libres. Dans la figure (A), les sous-figures de gauche et de droite représentent respectivement les filtres dans le domaine temporel et fréquentiel pour des filtres de tailles $L = 32$. La figure (B) est la même pour des tailles de filtres de $L = 256$. Comme on peut le voir, l'encodeur f_e apprend des formes d'ondes sinusoïdales de différentes fréquences et de différents décalages de phases. On peut ausi remarquer que les filtres de hautes fréquences sont très bruitées. Pour les filtres de la figure (B) on peut noter que des filtres à même empreinte fréquentielle ont des localisations temporelles très variées.

6.2.1 Inversibilité

Dans le cas de la séparation de sources, on veut en première priorité que le banc de filtres soit inversible. Si l'on prend comme front-end une couche de convolutions dont les filtres sont de tailles L et dont le pas d'avancement est de taille P, on doit avoir au moins P filtres pour permettre l'inversibilité de celui-ci. En effet, l'inversibilité dépend du pas d'avancement plutôt que de la taille des filtres. Par exemple, dans le cas trivial, on peut reconstruire parfaitement un signal quelconque en prenant P filtres composés d'un Dirac sur les P premiers échantillons de la fenêtre. Ainsi, un banc de filtres de rang au moins P est requis pour obtenir un banc de filtres pertinent pour permettre l'analyse et la synthèse de signaux. Dans le cas de Conv-TasNet, on utilise un pas d'avancement de taille $L/2$, il suffira donc d'avoir au minimum $L/2$ filtres pour s'assurer que le modèle puisse apprendre un banc de filtres inversible. En pratique, on

utilise un nombre de filtres C beaucoup plus grand que P.

6.2.2 Invariance aux décalages de phases

Si l'on superpose aléatoirement plusieurs sinusoïdes à différentes fréquences en tirant leur phase uniformément, alors la perception des différentes réalisations sera presque surement identique. Instinctivement, on voudrait donc construire des bancs de filtres capables d'encoder des décalages de phases d'une forme d'onde quelconque à travers la réponse d'un sous-ensemble de filtres sans modifier l'amplitude totale de cette réponse. Une définition précise de cette notion d'invariance à la phase sera formalisée plus loin. Dans le cas des filtres de la transformée de Fourier discrète, cette invariance à la phase est assurée par la présence de deux "sous filtres" de même fréquence, mais ayant un décalage de phase de $\pi/2$ (les parties cosinus et sinus du filtre). On peut définir ce même genre d'invariance à la phase pour des formes d'ondes quelconques à l'aide de la transformée de Hilbert.

Transformée de Hilbert

La transformée de Hilbert $\mathcal{H}(s_0)$ d'un signal s_0 est définie à partir de sa transformée de Fourier $S_0 = \mathcal{F}(s_0)$:

$$\forall f,\, \mathcal{F}\big(\mathcal{H}(s_0)\big)(f) = \begin{cases} S_0(f).e^{j\pi/2} & \text{si} f > 0, \\ S_0(f).e^{-j\pi/2} & \text{si} f < 0, \\ 0 & \text{sinon.} \end{cases} \tag{6.1}$$

Le signal temporel résultant $\mathcal{H}(s_0)(t)$ correspond donc à un signal ayant les mêmes contributions fréquentielles en amplitudes, mais avec un décalage de phase de $\pi/2$. On peut démontrer qu'un signal et sa transformée de Hilbert sont orthogonaux.

Représentation analytique du signal

À partir d'un signal $s_0(t)$ et de sa transformée de Hilbert $\mathcal{H}(s_0)(t)$, on peut construire le *signal analytique* $\tilde{s}_0(t)$:

$$\tilde{s}_0(t) = s_0(t) + j\mathcal{H}(s_0)(t), \tag{6.2}$$

qui s'écrit sous forme polaire :

$$\tilde{s}_0(t) = A_0(t)e^{i\varphi_0(t)}, \tag{6.3}$$

où $A_0(t) = \sqrt{s_0(t)^2 + \mathcal{H}(s_0)(t)^2}$ correspond à l'amplitude du signal (aussi appelée la partie modulante) et où $\varphi_0(t)$ correspond à la phase du signal, $e^{i\varphi_0(t)}$ peut être appelée partie modulée. On appellera filtre analytique un filtre complexe dont la partie imaginaire est la transformée de Hilbert de la partie réelle.

Transformée de Hilbert étendue

Dans le cadre de cette thèse, pour formaliser notre intuition de ce qu'est l'invariance à la phase d'un banc de filtres, nous définissons la transformée de Hilbert étendue $\mathcal{H}_\psi$ par :

$$\forall f,\ \mathcal{F}\big(\mathcal{H}_\psi(s_0)\big)(f) = \begin{cases} S_0(f).e^{j\psi} & \text{si} f > 0, \\ S_0(f).e^{-j\psi} & \text{si} f < 0, \\ 0 & \text{sinon.} \end{cases} \tag{6.4}$$

où ψ correspond à un décalage de phase quelconque entre 0 et 2π. Dans le cas où ψ est égal à $\pi/2$ on retombe sur la transformée de Hilbert. Notons que pour un signal analytique $\tilde{s}$ quelconque, on a l'égalité suivante :

$$\Re(\tilde{s}e^{i\psi})(t) = \mathcal{H}_\psi(s)(t) \tag{6.5}$$

Formalisation de l'invariance à la phase

Le produit scalaire entre un signal s et un filtre f_0 de longueur T est définie par :

$$\langle f_0, s\rangle = \sum_{t=0}^{T-1} f_0(t)s(t), \tag{6.6}$$

Pour un filtre analytique $\tilde{f_0}$, on obtient l'égalité suivante :

$$|\langle \tilde{f_0}, s\rangle| = |\langle \tilde{f_0}, \mathcal{H}_\psi(s)\rangle|, \tag{6.7}$$

où $\mathcal{H}_\psi(s)$ correspond à une version de s dont toutes les phases sont déplacées d'un facteur ψ. En construisant de tels filtres, on peut donc s'assurer que le banc de filtres sera invariant à des décalages de phases quelconques.

6.2.3 Invariance aux petits décalages temporels

Idéalement, on voudrait également que les filtres soient invariants à des décalages temporels, c'est-à-dire que pour des τ suffisamment petits, on ait :

$$|\langle \tilde{f_0}, s\rangle| = |\langle \tilde{f_0}, s_\tau\rangle|. \tag{6.8}$$

avec :

$$s_\tau(t) = s(t+\tau). \tag{6.9}$$

Si l'on se munit d'un filtre analytique $\tilde{f_0}$, l'invariance au décalage temporel est directement reliée à la forme de la partie modulante $A(t)$ du filtre. En effet, si $A(t)$ a des contributions hautes fréquences importantes, alors les oscillations induitent par celles-ci modifieront la réponse du produit scalaire pour des petites valeurs de τ.

Réponse fréquentielle de la partie modulante

En effet, en posant $f_0(t)$ un filtre réel quelconque et en prenant $\mathcal{H}(f_0)(t)$ sa transformée de Hilbert. On peut décomposer chacun de ces sous-filtres en série de Fourier tel que :

$$f_0(t) = \sum_k a_k \cos(2\pi\nu_k t + \psi_k), \tag{6.10}$$

$$\mathcal{H}(f_0)(t) = \sum_k a_k \sin(2\pi\nu_k t + \psi_k), \tag{6.11}$$

et, en prenant la partie d'amplitude A du filtre analytique résultant $\tilde{f_0}$ on peut précisément déterminer ses contributions fréquentielles en regardant :

$$A(t) = \sqrt{f_0(t)^2 + \mathcal{H}(f_0)(t)^2}, \tag{6.12}$$

$$A(t) = \sqrt{\Big(\sum_k a_k \cos\big(2\pi\nu_k t + \psi_k\big)\Big)^2 + \Big(\sum_k a_k \sin\big(2\pi\nu_k t + \psi_k\big)\Big)^2}, \tag{6.13}$$

$$A(t) = \sqrt{\sum_k a_k^2 + \sum_{k_1,k_2|k_1\neq k_2} a_{k_1} a_{k_2} \cos\big(2\pi(\nu_{k_1} - \nu_{k_2})t + (\psi_{k_1} - \psi_{k_2})\big)}, \tag{6.14}$$

enfin, en appliquant un développement limité d'ordre 1 on obtient :

$$A(t) \approx \sqrt{\sum_k a_k^2} + \frac{1}{2\sqrt{\sum_k a_k^2}} \sum_{k_1,k_2|k_1\neq k_2} a_{k_1} a_{k_2} \cos\big(2\pi(\nu_{k_1} - \nu_{k_2})t + (\psi_{k_1} - \psi_{k_2})\big). \tag{6.15}$$

On peut voir que la transformée de Fourier de $A(t)$ peut principalement être décomposée en sinusoïdes de fréquences $\nu_{k_1} - \nu_{k_2}$ (les autres composantes fréquentielles auront un impact beaucoup plus faible dans le domaine fréquentiel).

Critère d'invariance temporelle

Ainsi, si le support fréquentiel du filtre de base $f_0(t)$ est trop important, l'amplitude $A(t)$ du filtre analytique associé oscillera rapidement. Or, les oscillations hautes fréquences dans l'amplitude du filtre ne sont pas désirables pour obtenir des invariances aux décalages temporels.
Il serait donc utile de contraindre le support fréquentiel du filtre f_0 avant de prendre sa transformée de Hilbert pour s'assurer que les composantes fréquentielles (les différences $\nu_{k_1} - \nu_{k_2}$) des parties d'amplitude soient basses fréquences.

6.2.4 Sur-parametrisation

En prenant un filtre et sa transformée de Hilbert, on obtient un filtre complexe invariant en amplitude pour les décalages de phases (ie décalage de phase dans la partie modulée du signal sans modification de l'enveloppe dans la partie modulante). On pourrait donc obtenir cette propriété en complétant un banc de filtres appris par les transformées de Hilbert de ses filtres comme cela est fait dans [79]. Cependant, l'utilisation de réseau de neurones à valeurs réelles ne permet pas de prendre en compte convenablement l'amplitude induite par un filtre analytique. En effet, dans le cas des modèles de types Tas-Net, les parties réelle et imaginaire d'un filtre analytique sont calculées et traitées de manière indépendante.

Approximation de l'amplitude et de la phase

Guth et al. [90] montrent que pour un filtre réel f dont on note f_ψ la version déphasée de ψ et $\tilde{f}$ sa version analytique, on a :

$$|x \circledast \tilde{f}| = \frac{1}{2}\int_0^{2\pi} \mathrm{ReLU}\big(x \circledast f_\psi\big)\mathrm{d}\psi, \tag{6.16}$$

et, un réseau de neurones peut faire une bonne approximation de cette intégrale à partir de :

$$|x \circledast \tilde{f}| \approx \frac{\pi}{K}\sum_{k=0}^{K-1} \mathrm{ReLU}\big(x \circledast f_{\psi_k}\big),\ \psi_k = \frac{2\pi k}{K}, \tag{6.17}$$

pour $K > 4$, on peut déjà avoir une approximation correcte de cette amplitude [90]. Une approximation plus grossière mais pratique est :

$$|x \circledast \tilde{f}| \approx |\max_{\psi_k}\big(x \circledast f_{\psi_k}\big)|,\ \psi_k = \frac{\pi k}{K}, \tag{6.18}$$

De cette manière, le filtre s_{ψ_m} qui maximise la réponse du produit scalaire avec x donne simultanément une estimation de l'amplitude et une approximation de la phase ψ_m de la forme d'onde analysée.
Ainsi, construire un banc de filtres mettant en jeu le même filtre avec plusieurs décalages de phases semble être une piste intéressante pour sur-paramétriser l'encodeur d'un modèle de type TasNet. De cette manière, l'amplitude et la phase peuvent être codées dans un nombre réel (l'indice du filtre correspondrait ainsi à une approximation de la phase et sa valeur à l'amplitude correspondante). On suppose que c'est surtout cette accumulation de décalage de phase qui permet à un banc de filtres d'être expressif pour un réseau de neurones à valeurs réelles.
Dans leur article, Ditter et al. [76] utilise des filtres de gammatones fixés (sans apprentissage) et arrivent à obtenir des performances supérieures à l'utilisation de filtres libres appris [48] sur une tâche de séparation de sources voix-voix.

6.2.5 Localisation temporelle

Les convolutions comme on les utilise souvent dans les réseaux de neurones, c'est-à-dire avec un pas d'avancement important comparé à la taille de la fenêtre (*strided-convolution*), ne permet pas d'obtenir une bonne localisation temporelle.
Or, localiser précisément un évènement dans une fenêtre est nécessaire sur des tâches où l'on cherche à reconstituer du signal.
Expérimentalement, on s'aperçoit que les filtres réels appris par l'encodeur d'un CTN forment des ensembles de filtres ayant la même réponse fréquentielle, mais avec des positions différentes dans la fenêtre considérée.
Ces différentes positions permettent d'obtenir une meilleure précision temporelle tout en s'adaptant à la fréquence considérée lors de l'analyse à la manière d'une transformée en ondelettes.

6.3 Banc de filtres paramétrés

6.3.1 Encodeurs paramétrés

Afin de satisfaire les différentes propriétés présentées dans la section précédente, on peut ajouter une contrainte de structure aux filtres lors de l'apprentissage. On parle alors de banc de filtres paramétrés. Ces paramétrisations peuvent prendre plusieurs formes :

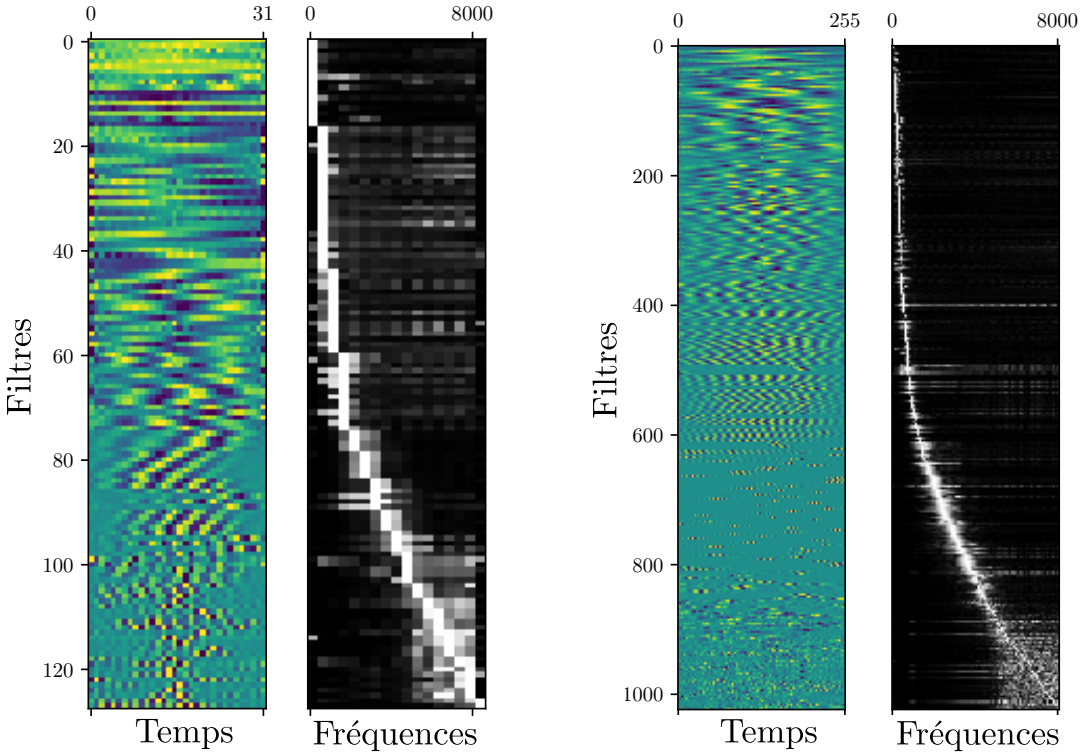
— soit par l'apprentissage de paramètres explicites sur une famille de filtres prédéfinies (comme la fréquence centrale, la largeur de bande, etc...). Cette approche limite le nombre de paramètres à apprendre.
— soit par l'induction d'un ensemble de sous-filtres calculés pour compléter un filtre appris librement de telle manière que cette famille exhibe globalement de bonnes propriétés de représentation.

On peut aussi combiner les deux approches afin de diminuer le nombre de paramètres et d'assurer l'obtention d'un maximum de bonnes propriétés sur le banc de filtres final.

Famille de filtres paramétrés

On présente ici des familles de bancs de filtres paramétrés utilisés dans un contexte d'apprentissage pour de la séparation de sources. On présentera également les propriétés émergeant naturellement de ces familles de bancs de filtres.

Sinc : Ce type d'encodeur a été proposé pour une tache d'ASR dans l'article SincNet de Ravanelli et al. [77]. Ce banc de filtres a également été étendu à des tâches de séparation de sources par Pariente et al [79]. Ces filtres sont construits à partir de sinus cardinaux (sinc). La réponse en fréquence des sinus cardinaux correspond à des fenêtres rectangulaires centrées en 0. En prenant la différence de deux sinus cardinaux,



(A) Kernel size 32 (B) Kernel size 256

FIGURE 6.2 – Banc de filtres analytiques appris. Dans la figure (A), les sous-figures de gauche et de droite représentent respectivement les filtres dans le domaine temporel et fréquentiel pour des filtres de tailles $L = 32$. La figure (B) est la même pour des tailles de filtres de $L = 256$. Comme on peut le voir, l'encodeur f_e apprend des formes d'ondes sinusoïdales de différentes fréquences et de différents décalages de phases. On peut ausi remarquer que les filtres de hautes fréquences sont très bruités. Pour les filtres de la figure (B) on peut noter que des filtres à même empreinte fréquentielle ont des localisations temporelles très variées.

on obtient un filtre passe-bande s_0 :

$$s_0(t|\nu_1, \nu_2) = 2\nu_2 \text{sinc}(2\pi\nu_2 t) - 2\nu_1 \text{sinc}(2\pi\nu_1 t), \tag{6.19}$$

la largeur de bande b et la fréquence centrale ν_0 sont déterminées par les fréquences de coupures ν_1 et ν_2 des deux sinus cardinaux avec $\nu_0 = \frac{\nu_1+\nu_2}{2}$ et $b = \frac{\nu_2-\nu_1}{2}$. Notons que ces filtres sont réels et ne permettent pas de représenter la phase. Ravanelli et al. réussissent à obtenir de bonnes performances en ASR car le banc de filtres est appliqué avec un pas d'avancement de 1 et la réponse en amplitude peut y être mesurée à l'aide d'une fonction de pooling. Dans le cas de la séparation de sources, l'utilisation d'un pas d'avancement de 1 entraîne un coût de calcul prohibitif pour les modèles de type Tas-Net. Avec cette première formulation et en prenant un pas d'avancement de moitié de la taille des fenêtres, Pariente et al. [79] obtiennent logiquement de mauvaises performances comparées à l'utilisation de filtres libres. En effet, l'analyse et la synthèse de signaux

est impossible si la phase de tous les filtres est identique. Ces considérations montrent que le banc de filtres est inconsistant dans un cadre de modèle de masquage. Dans le même article, les auteurs arrivent à rendre expressif pour la tache de séparation sources un banc de filtres constitué de sinus cardinaux et de leurs transformées de Hilbert.

Gabor : D'autres types de paramétrisations ont également été proposées pour des front-ends en ASR, comme les filtres de Gabor [78]. Ces derniers sont directement définis dans le domaine complexe et permettent une meilleure localisation temps-fréquence que les filtres *sinc*. Ces derniers sont paramétrés par leurs largeurs de bandes σ et leurs fréquences centrales ν_0 :

$$g(t|\nu_0, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} e^{2\pi j \nu_0 t}. \tag{6.20}$$

Contrairement au banc de filtres sinc, ce dernier est directement adaptable à la séparation de sources de part sa nature complexe. Cependant, on peut encore ajouter de l'expressivité à ce banc de filtres en apprenant également un terme de phase ψ dans la sinusoïde des filtres :

$$g(t|\nu_0, \sigma, \psi) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \cos(2\pi\nu_0 t + \psi). \tag{6.21}$$

Comme expliqué dans la partie 6.2.4, la multiplication des décalages de phase aide le modèle à correctement déterminer l'amplitude et la position de la forme d'onde étudiée.

Gammatone : La superposition de multiples décalages de phase dans les filtres de l'encodeur a été particulièrement étudiée par Ditter et al. dans leur article sur les bancs de filtres Gammatones [76] (*Multi-Phase Gammatone FilterBank* ou MPGTF dans la suite). L'idée principale de leur article est de fixer le banc de filtres sur des Gammatones et de prendre un certain nombre de décalages de phase en fonction de la fréquence fondamentale du filtre. Le MPGTF se décompose comme suit :

$$\gamma(t|a, b, p, \nu_0, \psi_i) = at^{p-1} e^{-2\pi bt} \cdot \cos(2\pi\nu_0 t + \psi_i), \tag{6.22}$$

où a est un paramètre de normalisation, p l'ordre du filtre, b la largeur de bande, ν_0 la fréquence centrale et ψ_i la phase du filtre. Les ν_0 sont fixées et distribuées sur une échelle ERB [91]. Le paramètre de phase ψ_i est choisi pour construire plusieurs filtres ayant la même fréquence centrale ν_0 et la même enveloppe $at^{n-1}e^{-2\pi bt}$ mais avec des valeurs ψ distribuées linéairement sur $[0, 2\pi]$. Le nombre de filtres utilisés pour un ν_0 donné dépendra de sa fréquence. En effet, pour les basses fréquences, la discrimination de la phase est plus importante que pour les hautes fréquences (où deux décalages de phases suffisent à couvrir tous les cas). Dans leur article, les paramètres des filtres sont fixés, mais le modèle résultant arrive tout de même à obtenir des performances supérieures à l'utilisation de filtres libres. Des travaux ultérieurs [14] ont tenté de paramétrer ces filtres afin que le réseau puisse apprendre sa propre échelle. Cependant, les résultats de ces travaux n'ont pas montré une grande amélioration sur les performances.

Autres paramétrisations Il existe d'autres paramétrisations qui ne seront pas approfondies ici. Les trois types de paramétrisations présentées ci-dessus sont suffisamment généraux pour distinguer les différentes propriétés qu'un banc de filtres paramétrés peut avoir. On pourra cependant trouver d'autres filtres de bases dans l'article [92] comme les ondelettes "mexican-hat" ou les filtres "gammachirps".

Paramétrages implicites

Front-end analytique Pariente et al. [79] proposent de construire un banc de filtres en se basant sur la transformée de Hilbert afin de contraindre directement l'invariance au décalage de phase sur des filtres quelconques. En posant un filtre s_0 quelconque appris librement, ils induisent un autre filtre $\mathcal{H}(s_0)$. De cette manière, le modèle a beaucoup plus de liberté dans la nature des filtres qu'il peut apprendre. Expérimentalement, ils montrent que leur banc de filtres, dit analytique, permet à la fois d'augmenter les performances, mais aussi d'augmenter largement la taille des fenêtres.

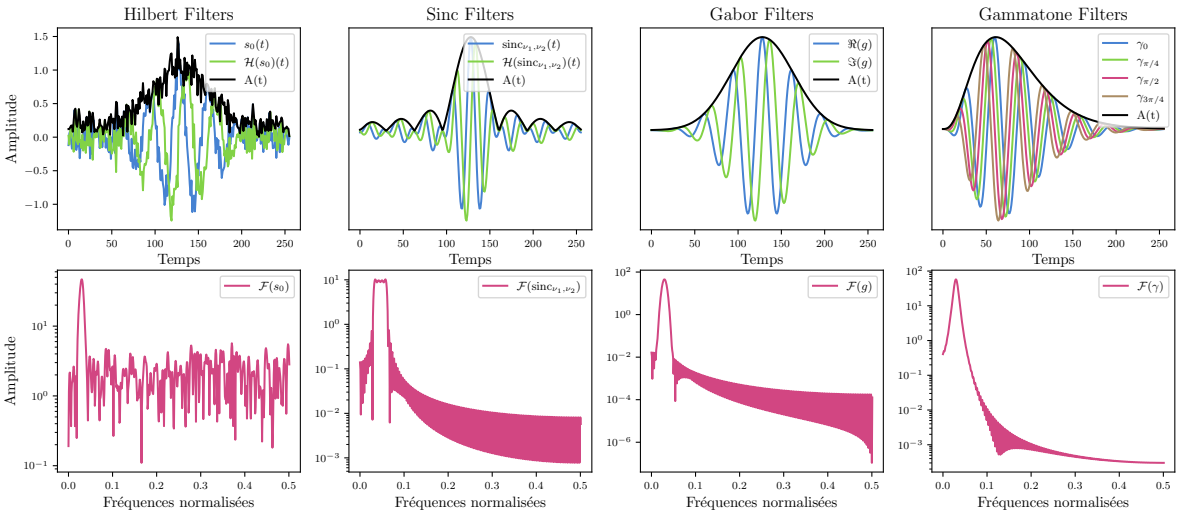


FIGURE 6.3 – Exemples de filtres de Hilbert, Sinc analytique, Gabor complexe et Multi-phase Gammatone.

Éléments de comparaisons

Dans le tableau 6.1, on référence l'ensemble des propriétés des filtres présentés jusqu'ici. La figure 6.3 illustre également des exemples de ceux-ci. Comme on peut le voir, aucune des paramétrisations ne permet de garantir l'ensemble des propriétés évoquées jusqu'ici. Cependant, les paramétrisations libres et analytiques permettent potentiellement d'obtenir toutes ces propriétés une fois l'apprentissage effectué.

TABLE 6.1 – Résumé des différentes propriétés liées à chaque banque de filtres. 💡 indique la capacité des bancs de filtres à apprendre ces propriétés.

Encoder	Learnable	ϕ-invariant	τ-invariant	ϕ-surparams	t-localisation
Real free [48]	✓	💡	💡	💡	💡
Hilbert free [79]	✓	✓	💡	💡	💡
Sinc [77]	✓	✗	✗	✗	✗
Hilbert Sinc [79]	✓	✓	✓	✗	✗
Gabor [78]	✓	✓	✓	✗	✗
Gabor$_\phi$	✓	💡	💡	💡	✗
MPGTF [76]	✗	✓	✓	✓	✗
STFT	✗	✓	✓	✗	✗
Random	✗	✗	✗	✗	✗

6.4 Contributions

Comme mis en avant dans le tableau 6.1, aucun des frontends présentés jusqu'ici ne permet de prendre en compte toutes les propriétés de manière naturelle. La principale difficulté est de construire des bancs de filtres suffisamment diversifiés en temps et en fréquence tout en assurant les invariances de décalages de phase et de temps. Dans cette partie, on va se concentrer sur la construction d'un banc de filtres permettant à la fois d'obtenir des représentations étant par nature :

— Invariantes aux décalages de phase (propriétés d'analycité, de surparamétrisation de phase),
— Invariantes aux décalages temporelle (enveloppe basse fréquence),
— Et permettant l'apprentissage d'une localisation temporelle et fréquentielle la moins contraignante possible.

6.4.1 Banc de filtres polyphases

Afin de généraliser la surparamétrisation de phase présente dans le front-end MPGTF, on propose ici d'étendre la transformée de Hilbert à d'autres valeurs de phase que $\frac{\pi}{2}$. Pour cela, il suffit de décaler la phase d'un filtre de base $s_0(t)$ vers une phase désirée ψ en appliquant la transformation suivante dans le domaine fréquentiel :

$$\mathcal{F}\big(\mathcal{H}_\psi(s_0)\big)(f) = \begin{cases} S_0(f).e^{j\psi} & \text{si} f > 0, \\ S_0(f).e^{-j\psi} & \text{si} f < 0, \\ 0 & \text{sinon.} \end{cases} \tag{6.23}$$

Ainsi, on ré-utilise :

— L'idée de la transformée de Hilbert (décaler la phase d'un filtre quelconque tout en conservant sa réponse en fréquence),
— et l'idée des MPGTF (ensemble de décalages de phase ψ_i des filtres de l'équation 6.22).

À partir d'un filtre de base $s_0(t)$, on peut déduire un ensemble de K filtres :

$$s_k(t) = \mathcal{H}_{k\pi/K}(s_0)(t), \quad k \in \{0, \ldots, K-1\}. \tag{6.24}$$

Ceci correspond à échantillonner le demi-cercle unitaire supérieur du plan complexe. En effet, si l'on décide de ne pas appliquer de non-linéarité après le banc de filtres, échantillonner le cercle complet apporterait une information identique sur certains décalages de phase à un facteur -1 près. Ce facteur étant un nombre réel, un réseau de neurones à valeurs réelles n'aura aucune difficulté à en tirer de l'information.
Un effet secondaire mais bénéfique de cette méthode, est qu'elle réduit le nombre de filtres de base à apprendre d'un facteur K.
De plus, à l'aide de cette décomposition, on peut aussi décrire les K filtres induit par s_0 comme le produit d'une partie modulante et d'une partie modulée :

$$s_k(t) = A(t) \cdot \cos\left(\phi(t) + \frac{k\pi}{K}\right), \quad k \in \{0, \ldots, K-1\}, \tag{6.25}$$

Cette réécriture permet de généraliser la sur-paramétrisation de phase, et permet (comme on le verra plus loin au moyen d'expériences numériques) de conserver les performances de l'encodeur analytique tout en diminuant le nombre de paramètres d'un facteur K. Cependant, ce banc de filtres souffre des mêmes problèmes que le banc de filtres analytiques. En effet, la partie modulante $A(t)$ peut ne pas être basse fréquence. Les filtres ainsi obtenus restent bruités.
Ce bruit augmente à mesure que la taille des fenêtres augmente. Il altère peu ou prou la réponse du filtre, ce qui peut en même temps entraîner des erreurs lors de la construction du masque par le séparateur et empêcher l'invariance aux décalages temporels.
Afin de s'assurer que ces deux points soient corrigés, il faut donc contraindre le filtre de base s_0 à avoir sa partie modulante $A(t)$ basse fréquence.

6.4.2 Front-end de Bedrosian

Dans cette partie, nous cherchons à construire un filtre s_0 le plus libre qui soit tout en assurant qu'il soit analytique et invariant aux décalages temporels. Partons d'un filtre s_0 quelconque se décomposant en une partie modulante et une partie modulée :

$$s_0(t) = A(t).\cos(\phi(t)). \tag{6.26}$$

Rappelons que l'objectif premier de la transformée de Hilbert est de construire deux filtres orthogonaux ayant les mêmes composantes fréquentielles afin de donner un sens à la phase pour des filtres arbitraires. Cette décomposition modulant-modulé est donc assurée quelque soit le filtre s_0 en prenant l'amplitude et la phase de sa forme analytique $\tilde{s}_0(t) = s_0(t) + i\mathcal{H}(s_0)(t)$.

Factorisation

Jusqu'ici, on a déduit les parties $A(t)$ et $\cos(\phi(t))$ grâce à la transformée de Hilbert. Cependant, comme on l'a vu précédemment, ce critère n'est pas suffisant pour induire de bonnes propriétés sur les filtres. Plus particulièrement, dans le cas où l'on voudrait explicitement construire une enveloppe basses fréquences ayant un maximum de degrés de libertés, on voudrait construire l'enveloppe du filtre dans un premier temps avant de trouver une partie modulée qui convienne.
On cherche donc à construire un couple de fonctions $A(t)$ et $\phi(t)$.
Le théorème de Bedrosian [93] donne un critère suffisant sur la structure de $A(t)$ et $\cos(\phi(t))$ pour garantir l'analycité du filtre complexe résultant.
Soit f, g deux fonctions à valeurs réelles dans $L^2(\mathcal{R})$, si les supports de leur transformée de Fourier sur $\mathbb{R}^+$ sont des compacts disjoints[1], avec f de support basses fréquences et g de support hautes fréquences, alors le théorème de Bedrosian stipule que :

$$\mathcal{H}(fg)(t) = f\mathcal{H}(g). \tag{6.27}$$

En associant f à $A(t)$ et g à $\cos(\phi(t))$, l'équation 6.27 se réécrit :

$$\mathcal{H}(A\cos(\phi))(t) = A(t)\mathcal{H}(\cos(\phi))(t) = A(t)\sin(\phi(t)). \tag{6.28}$$

En d'autres termes, le signal suivant $\tilde{s}_0$ est analytique :

$$\tilde{s}_0 = A(t)e^{j\phi(t)}. \tag{6.29}$$

On peut donc construire les parties modulante et modulée du filtre de manière indépendante à la seule condition que leur support fréquentiel ne se chevauche pas. $A(t)$ étant identifié à un signal basse fréquence et $\cos(\phi(t))$ à un signal haute fréquence, on obtient ce que l'on veut.

Détails d'implémentations

Regardons maintenant une méthode de paramétrisation pour assurer la contrainte mise sur les supports des deux parties modulante et modulée des filtres. On peut construire la partie modulée $\cos(\phi(t))$ en utilisant une simple sinusoïde paramétrée par sa fréquence ν_0 comme dans le cas des filtres de Gabor :

$$\cos(\phi(t)) = \cos(2\pi\nu_0 t). \tag{6.30}$$

Ce choix est également motivé par la discussion autour de l'équation 6.12 qui renforce l'idée de contraindre la partie modulée à avoir un support de fréquentiel restreint.
Pour satisfaire le critère de support pour pouvoir appliquer le théorème de Bedrosian, il faut donc simplement contraindre la partie modulante à avoir un support fréquentiel dans $[-\nu_0, \nu_0]$. On peut donc construire $A(t)$ en apprenant un filtre réel $a(t)$ quelconque

1. Les supports de f et g doivent donc être de la forme $[0, a]$ et $[b, +\infty[$ avec $a \leq b$.

que l'on convolue ensuite avec un filtre passe-bas gaussien g. Ce passe-bas est construit de telle sorte à avoir une atténuation de -20 dB en ν_0. Ce choix est fait afin d'avoir des modifications lisses sur $A(t)$ puisque la valeur de ν_0 est vouée à changer durant l'apprentissage. On peut donc décrire la partie modulante ainsi :

$$A(t) = a(t) \circledast g(t), \tag{6.31}$$

$$g(t) = e^{-\frac{t}{\sigma_{\nu_0}}^2}, \tag{6.32}$$

$$\tag{6.33}$$

avec σ_{ν_0} choisie pour que le filtre passe-bas g ait une atténuation de -20dB en ν_0 dans le domaine fréquentiel. Il est en fait plus simple d'apprendre directement la partie modulante $A(t)$ dans le domaine fréquentiel en apprenant un filtre $\mathcal{F}(a)$ et en prenant la transformée de Fourier inverse du produit :

$$\mathcal{F}(A)(\nu) \propto \mathcal{F}(a)(\nu) \cdot e^{-(\sigma_{\nu_0}\nu)^2}, \tag{6.34}$$

La contrainte de positivité sur A est quant à elle obtenue en prenant le filtre final A^+ :

$$A^+(t) = A(t) - \min_t(A(t)). \tag{6.35}$$

Pour des décalages de phases arbitraires, il n'est pas nécessaire d'utiliser la transformée de Hilbert étendue. En effet, puisque que l'on connaît la fonction $\phi(t) = 2\pi\nu_0 t$, il suffit de la déphaser d'une valeur ψ quelconque tel que :

$$\phi_\psi(t) = 2\pi\nu_0 t + \psi. \tag{6.36}$$

L'ensemble de filtres $(s_k)_{k<K}$ induits par les parties modulantes et modulées peuvent finalement s'écrire :

$$s_k = A^+(t)\cos(2\pi\nu_0 t + \frac{k\pi}{K}). \tag{6.37}$$

Complément : vers des parties modulées génériques

Les filtres construits dans la partie précédente ne permettent pas de représenter des fonctions $\phi(t)$ plus complexes que de simples fonctions linéaires (et donc de simples sinusoïdes). Seulement, il faut réussir à trouver une contrainte à appliquer sur $\phi(t)$ pour assurer que le support fréquentiel de $\cos(\phi(t))$ ne déborde pas sur celui de la partie d'amplitude $A(t)$.

Meyer et al. [94] donnent deux conditions suffisantes sur $A(t)$ et $\phi(t)$ pour assurer que l'oscillation induite par $\phi(t)$ ne perturbe pas l'amplitude $A(t)$:

$$\left|\frac{\dot{A}(t)}{A(t)\dot{\phi}(t)}\right| \ll 1, \tag{6.38}$$

$$\left|\frac{\ddot{\phi}(t)}{\dot{\phi}(t)}\right| \ll 1, \tag{6.39}$$

P. Flandrin et al. [95] présentent l'intuition suivante pour expliquer ces deux inégalités : En considérant la pseudo période $T(t) = 2\pi/\dot{\phi}(t)$, la première inégalité indique que l'amplitude $A(t)$ ne doit pas subir de variation importante dans une pseudo-période. La deuxième indique que $T(t)$ doit être lentement variable, c'est-à-dire que $\cos(\phi(t))$ doit avoir le temps de se dérouler avant de changer de fréquence instantanée.
En satisfaisant ces deux contraintes, on pourrait décrire des filtres dont la fréquence instantanée ne se limite pas à une constante, par exemple des chirps.
Dans la discussion suivante, on étudiera la fonction $\varphi(t)$:

$$\varphi(t) = \frac{\phi(t)}{2\pi} \tag{6.40}$$

Pour se faire, on se munit d'un nombre restreint de points de contrôle représentant la fréquence instantanée $\dot{\varphi}(t)$ à des pas de temps donnés. Pour une fenêtre de taille L, on se donne P points de contrôle p_i placés linéairement sur L en t_{p_i} et de fréquence normalisée ν_{p_i}. Il suffit ensuite de faire une interpolation linéaire de ces ν_{p_i} sur L afin de construire $\dot{\varphi}(t)$ sur tout l'intervalle $[0, L]$. Enfin, on peut simplement construire $\varphi(t)$ en intégrant $\dot{\varphi}(t)$ sur L :

$$\varphi(t) = \sum_{\tau=0}^{t} \dot{\varphi}(\tau). \tag{6.41}$$

On illustre ces différentes étapes de constructions dans les figures 6.4 et 6.5.
L'interpolation linéaire sur L de ces ν_{p_i} donne :

$$\dot{\varphi}(t) = \frac{P}{L}(\nu_{p_{i+1}} - \nu_{p_i})t + c, \; \forall t \in [t_{p_i}, t_{p_{i+1}}], \tag{6.42}$$

$$c = \nu_{p_i} - \frac{P}{L}(\nu_{p_{i+1}} - \nu_{p_i})t_{p_i}, \tag{6.43}$$

$$\ddot{\varphi}(t) = \frac{P}{L}(\nu_{p_{i+1}} - \nu_{p_i}), \; \forall t \in [t_{p_i}, t_{p_{i+1}}], \tag{6.44}$$

$$\tag{6.45}$$

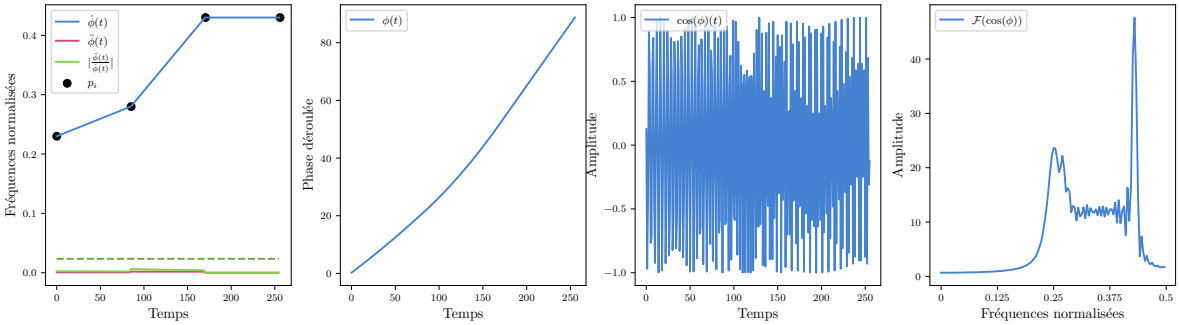


FIGURE 6.4 – Exemple de partie modulée $\cos(2\pi\varphi(t))$ construite à l'aide de la stratégie des points de contrôles (4 points de contrôle) avec $\nu_{min} = .2$ et $\nu_{max} = .5$. La ligne en pointillés verts correspond à la borne de l'équation 6.51. On peut voir que les fonctions $\ddot{\varphi}(t)$ et $|\frac{\ddot{\varphi}(t)}{\dot{\varphi}(t)}|$ prennent des valeurs de l'ordre de 10^{-3}.

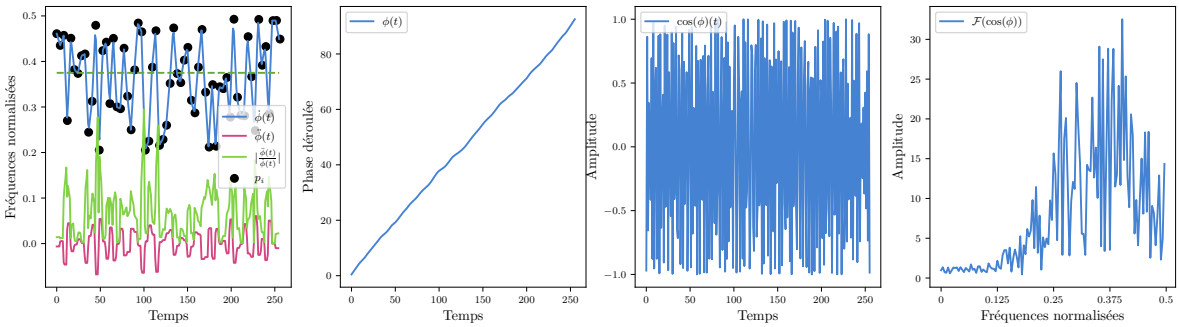


FIGURE 6.5 – Exemple de partie modulée $\cos(2\pi\varphi(t))$ construite à l'aide de la stratégie des points de contrôles (64 points de contrôle) avec $\nu_{min} = .2$ et $\nu_{max} = .5$. La ligne en pointillés verts correspond à la borne de l'équation 6.51. Dans ceci, les fonctions $\ddot{\varphi}(t)$ et $|\frac{\ddot{\varphi}(t)}{\dot{\varphi}(t)}|$ prennent des valeurs de l'ordre de 10^{-1}. On peut également voir que le support fréquentiel de $\cos(2\pi\varphi)$ n'est plus borné.

En notant $\delta_{p_i} = \frac{P}{L}\left|\nu_{p_{i+1}} - \nu_{p_i}\right|$, on a donc :

$$\left|\frac{\ddot{\varphi}(t)}{\dot{\varphi}(t)}\right| = \left|\frac{\delta_{p_i}}{\delta_{p_i} t + \nu_{p_i} - \delta_{p_i} t_{p_i}}\right|, \tag{6.46}$$

$$= \left|\frac{\delta_{p_i}}{\delta_{p_i}\left(t + \nu_{p_i}/\delta_{p_i} - t_{p_i}\right)}\right|, \tag{6.47}$$

$$= \left|\frac{1}{t - t_{p_i} + \nu_{p_i}/\delta_{p_i}}\right|, \tag{6.48}$$

et en remarquant que $t - t_{p_i} \geq 0$ on a :

$$\left|\frac{\ddot{\varphi}(t)}{\dot{\varphi}(t)}\right| \leq \left|\frac{1}{\delta_{p_i}/\nu_{p_i}}\right|, \tag{6.49}$$

$$\leq \frac{P}{L}\frac{\left|\nu_{p_{i+1}} - \nu_{p_i}\right|}{\nu_{p_i}}, \tag{6.50}$$

$$\leq \frac{P}{L}\frac{\nu_{max} - \nu_{min}}{\nu_{min}}, \tag{6.51}$$

Ainsi, en choisissant un nombre de points de contrôle P suffisamment faible devant L et en faisant attention aux choix des ν_{max} et ν_{min}, on assure de satisfaire la contrainte de l'équation 6.39. On peut donc construire des fonctions $\varphi(t)$ arbitraires tout en s'assurant de contraindre le support fréquentiel une fois le cosinus appliqué.
Notons que pour des nombres de points de contrôles faibles, on obtient les cas particuliers suivants :

— si $P = 1$: on se retrouve dans le cas où φ est une fonction linéaire,
— si $P = 2$: on modélise un chirp allant de la fréquence ν_{p_0} à la fréquence ν_{p_1}.

6.4.3 Réduction du coût computationel

On peut obtenir des simplifications lors de la projection d'un signal sur les bases de filtres de Hilbert étendues ou de Bedrosian en notant que ces filtres sont des combinaisons linéaires les uns des autres. En effet, pour un filtre s_0 et sa forme analytique $\tilde{s}_0 = s_0 + i\mathcal{H}(s_0)$, on peut calculer un décalage de phase s_ψ quelconque en prenant :

$$s_\psi = \Re\mathrm{e}\big(\tilde{s}_0 e^{i\psi}\big),\ \forall\psi \in \Psi. \tag{6.52}$$

Donc pour un signal x on a :

$$\langle x, s_\psi\rangle = \Re\mathrm{e}\big(\langle x, \tilde{s}_\psi\rangle\big) \tag{6.53}$$

$$= \Re\mathrm{e}\big(\langle x, \tilde{s}_0 e^{i\psi}\rangle\big) \tag{6.54}$$

$$= \Re\mathrm{e}\big(\langle x, \tilde{s}_0\rangle e^{i\psi}\big). \tag{6.55}$$

Ainsi, il suffit simplement de calculer la convolution du signal entre la partie réelle et la partie imaginaire du filtre pour automatiquement déterminer la réponse de tout les autres décalages de phases. En fonction du nombre K de décalages de phases on peut économiser un facteur $2/K$ en coût de calcul dans l'encodeur.

6.5 Résultats expérimentaux

6.5.1 Protocole expérimental

Dans la suite, on compare les différents front-ends (pré-existants et proposés) sur plusieurs configurations de modèles sur deux tâches de séparation de sources :

Dataset

Séparation de sources voix-voix Pour cette partie, on utilise le dataset LibriMix [61], un dataset open-source pour la séparation voix-voix. L'entraînement est effectué sur le sous-ensemble *train-clean-360* et les mélanges sont construits dynamiquement durant l'apprentissage pour une durée de 3 secondes.

Séparation de sources voix-bruits Pour cette tâche, on utilise le dataset du challenge *Deep Noise Suppression* [67] (DNS). Comme pour le cas voix-voix, les mélanges sont effectués dynamiquement sur une durée de 3 secondes.

Modèles

Les séparateurs f_s utilisés dans les expériences qui suivent sont des *Temporal Convolutional Network* (TCN). Les différentes expériences verront varier les encodeurs f_e selon le type de front-end, mais aussi selon différentes hyper-paramétrisations de ceux-ci sur la taille des fenêtres ainsi que sur le nombre de filtres utilisés. Typiquement, un TCN est une succession de R enchaînement de X blocs de convolutions dilatées successives. Les filtres des convolutions utilisés dans cette architecture sont de tailles P. Chaque bloc est composé d'une succession de couches linéaires de taille H, d'une convolution séparable ayant H canaux suivis de deux couches linéaires [48] (une pour les résidus de S canaux et une comme *bottleneck* de B canaux).

Hyper-paramétrisation L'hyperparamétrisation du séparateur sera quant à lui variable en fonction de l'hyper-paramétrisation des encodeurs. Le nombre de répétitions, de convolutions et le nombre de canaux seront modulés afin d'avoir un nombre de FLOPS équivalent sur tous les modèles. La table 6.2 présente les différents hyper-paramètres sélectionnés.
Notons que les hyper-paramètres des TCN étudiés ne sont pas optimaux, l'objectif ici est surtout d'identifier les tendances relatives aux propriétés des encodeurs.

Entraînement Tous les modèles sont entraînés en utilisant un optimiseur Adam [69] avec un taux d'apprentissage de 10^{-3} sur une fonction de coût Si-SNR 4.3.2 sur des signaux échantillonnés à 16kHz. Le taux d'apprentissage est divisé par deux toutes les 5 époques (1 époque = 20 000 signaux) si le score en validation n'augmente pas. L'entraînement d'un modèle prend approximativement deux jours sur des cartes graphiques 1080 Ti.

TABLE 6.2 – Hyper-paramétrisations des Conv-TasNet utilisés. N et L réfère respectivement aux nombres de filtres et à leurs tailles dans l'encodeur f_e. H, B, S, R, B et P correspondent aux hyper-paramètres du TCN.

	N	L	H	B	S	R	X	P	GFLOPS
Small $Sm.$	128	32	128	128	128	2	6	3	1.2
Mid $Mi.$	512	128	170	170	170	2	6	3	1.1
Large $Lg.$	1024	256	256	256	256	2	6	3	1
Very Large $Vl.$	2048	1024	512	512	512	2	6	3	1.3

TABLE 6.3 – Différence de performance sur un petit modèle Conv-TasNet (taille de filtres de $L = 32$ et nombre de filtres $N = 64$) pour une tâche de séparation de sources sur Librispeech. Gabor$_{real}$ utilise uniquement la partie réelle (cosinus) d'un filtre de Gabor, Gabor$_{complexe}$ utilise la partie réelle et la partie imaginaire (cosinus et sinus) et Gabor$_{\psi}$ utilise la partie réelle, mais utilise un paramètre ψ qui permet un déphasage du cosinus.

Encoder	Si-SNR
Gabor$_{real}$	5.91
Gabor$_{complexe}$	7.34
Gabor$_{\psi}$	7.82

6.5.2 Importance des décalages de phase

En guise d'analyse préliminaire sur l'importance de la sur-paramétrisation de phase, la table 6.3 présente les performances obtenues par trois modèles appris sur trois régimes de paramétrisation de la phase de banc de filtres de Gabor (présenté dans la partie 6.3.1) :

— Gabor$_{real}$: Banc de filtres constitué exclusivement de cosinus,
— Gabor$_{complexe}$: Banc de filtres constitué de partie cosinus et sinus,
— Gabor$_{\psi}$: Banc de filtres constitué exclusivement de cosinus dont la phase peut être modifiée par un paramètre ψ.

Ces résultats suggèrent que donner de la liberté à la phase des filtres aide effectivement le modèle à obtenir de meilleures performances.

6.5.3 Effet du nombre de décalages de phase K

Afin de comprendre l'impact sur la redondance des filtres couplés à leurs nombres de décalages de phase K, nous présentons ici un ensemble d'expériences effectuées sur des encodeurs de type transformée de Hilbert étendu. Pour cette expérience on prend des fenêtres de taille $L = 256$ avec un nombre de filtres N fixé à 1050 qui est divisible par $\{1, 2, 3, 5, 7, 10, 15\}$.

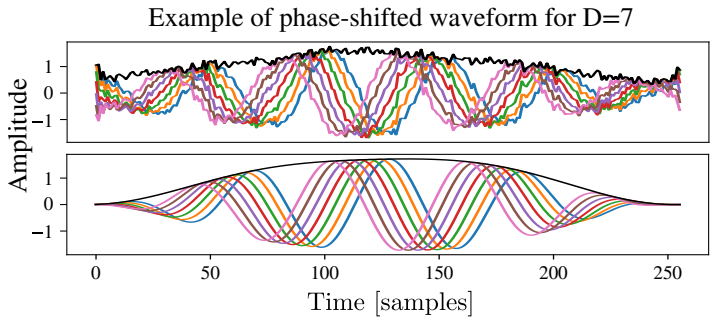


FIGURE 6.6 – [Haut] Exemple de filtre de base $s_0(t)$ et ses transformées de Hilbert étendues dans le cas $K = 7$. En noir, l'enveloppe temporel de cet ensemble de filtres. [Bas] Pareil avec des filtres de Bedrosian.

Le nombre de filtres de base appris N_w est donc égale à N/K. Notons que le cas $K = 1$ correspond à l'apprentissage d'un banc de filtres libres et que le cas $K = 2$ correspond à l'apprentissage de filtres analytiques (transformée de Hilbert).

TABLE 6.4 – Performance pour différents nombres de décalages de phases K pour la tâche de séparation de sources voix-voix sur la base de données LibriMix.

N_w	1050	525	340	210	**150**	105	70
K	1	2	3	5	**7**	10	15
SiSNR	10.11	10.66	10.61	10.64	**10.73**	10.45	9.37

Comme on peut le voir dans la table 6.4, il semble plus important d'avoir peu de filtres de base N_w avec une précision de phase importante que d'avoir beaucoup de filtres de bases décorrélés en phase. Cependant, augmenter le nombre de filtres de bases de manière trop importante $K \geq 10$ entraîne une chute des performances.
En effet, si le nombre de filtres de bases devient inférieur à $L/2 = 128$, le banc de filtres résultant n'est pas inversible.
Dans la figure 6.6, on présente un ensemble de filtres pour le cas $K = 7$. Comme on peut le voir, les filtres tendent à converger vers des sinusoïdes modulées, mais présentent encore une composante de bruit importante. Ces résultats motivent d'autant plus l'utilisation des front-ends de Bedrosian.

6.5.4 Performances sur les différents types de front-ends

Séparation de sources voix-voix

Nous comparons maintenant les différents front-ends préexistants (libre, Gammatone, analytique) avec les filtres de Hilbert étendus et de Bedrosian que nous proposons. Nous ajoutons également à la comparaison un front-end aléatoire fixé durant tout l'apprentissage. Nous testons deux hypothèses :

1. Notre première hypothèse est que lorsque L augmente, les performances des filtres non structurés (Random fix, Free) diminuent alors que celles des filtres

structurés (Analytique, Hilbert étendu et modèle de Bedrosian) ne diminuent pas. Nous comparons les résultats pour L=32 (Sm.), 128 (Mi.) et 256 (Lg.). Ceci illustre le fait que lorsque L augmente, la structure des filtres joue un rôle de plus en plus important. Le tableau 6.5 confirme notre hypothèse, il montre même une légère augmentation du SiSNR pour les filtres structurés (modèle analytique, Hilbert étendu et Bedrosian) pour cet hyper-paramétrisation. Comme nous l'avons vu, les résultats de Gammatone diminuent avec L car ils ne permettent pas de modéliser la localisation temporelle. Pour des L importants, les filtres de Bedrosian proposés obtiennent le meilleur Si-SNR (10,78). Ces résultats montrent qu'il est possible de gagner en interprétabilité tout en maintenant (voire en augmentant) les performances lorsque les filtres sont plus grands. Pour vérifier si nous pouvons encore étendre L, nous comparons les résultats obtenus avec des fénêtres très grandes ($Vl.$) L=1024. Dans ce cas, les performances de tous les front-ends chutent (en particulier celles du front-end libre).

2. Notre deuxième hypothèse est que les filtres structurés nécessitent moins de données d'apprentissage. Nous testons cette hypothèse dans le Tableau 6.6 en utilisant 0,1%, 1%, 10% ou 100% des données. Notre hypothèse n'est cependant pas confirmée : les performances de tous les frontends diminuent de façon similaire.

TABLE 6.5 – Performance (SiSNR) des différents encodeurs f_e.

Encodeur / L	$Sm.$	$Mi.$	$Lg.$	$Vl.$
Aléatoire fixé	9.85	9.53	9.08	7.12
Libre	10.40	10.33	10.11	7.76
Gammatone [76]	10.43	8.34	6.52	/
Hilbert [79]	10.31	10.41	10.66	8.61
Hilbert étendue	10.37	**10.51**	10.73	8.58
Bedrosian	**10.50**	10.43	**10.78**	8.47

TABLE 6.6 – Performance (SiSNR) des différents front-ends f_e en fonction de la quantité de données utilisées durant l'apprentissage dans le cas des fenêtres Lg.

Encodeur / Data	0.1%	1%	10%	100%
Libre	6.16	8.66	9.61	10.11
Hilbert [79]	6.15	8.72	10.12	10.66
Hilbert étendue	6.21	8.88	10.05	10.73
Bedrosian	6.14	8.61	9.97	10.78

TABLE 6.7 – Performance des différents front-ends pour de la séparation de sources voix-bruit.

Encodeur / L	*Sm.*	*Mi.*	*Lg.*
Aléatoire fixé	9.39	9.60	9.55
Libre	9.42	9.89	10.01
Sinc analytique [79]	9.57	9.81	9.95
Gabor	9.57	9.81	10.20
Hilbert [79]	9.67	9.98	10.53
Hilbert étendue	9.57	10.11	10.48
Bedrosian	9.48	10.03	10.52

Séparation voix-bruit

Dans le cas de la séparation voix/non-voix, il y a une plus grande différence entre les performances des petites et des grandes fenêtres. Cela peut s'expliquer par le fait que la voix et l'environnement sonore sont plus facilement identifiables sur des fenêtres plus grandes, puisque l'identification des sons harmoniques/non harmoniques est plus difficile sur les petites fenêtres que sur les grandes. De plus, de la même manière que pour la séparation voix-voix, nous observons que les filtres analytiques sont généralement plus performants que les filtres réels. Ceci est vrai pour les filtres paramétrés (gabor, sinc) et les filtres de Hilbert.

6.6 Conclusion

Dans ce chapitre, nous avons étudié les encodeurs/front-ends audio existants pour la séparation de sources et en avons proposé deux nouveaux front-ends : un établi sur la transformée de Hilbert étendue, et un autre établi à partir du théorème de Bedrosian. Ce dernier permet de gagner en interprétabilité des filtres tout en conservant des performances similaires et même en les améliorant légèrement dans le cas de filtres de grande taille. Ce paramétrage de front-ends permet de contourner le problème du bruit qui est généralement inséré dans les filtres appris ; il permet également de conserver une plus grande flexibilité que l'architecture SincNet [77] et pourrait potentiellement être utilisé pour d'autres tâches telles que l'ASR. De plus, pour le cas spécifique de la séparation de sources, la sur-paramétrisation de phase semble également être un point crucial pour construire des front-ends appliqués à des tâches d'analyse-synthèse dans le cas des réseaux de neurones à valeurs réelles.

Chapitre 7

Réseaux de neurones à convolutions paramétrées

L'objectif de cette partie est d'étendre les méthodes développées au chapitre 6 pour les front-ends à toutes les couches d'un réseau de neurones.

7.1 Introduction

Comme on a déjà pu l'expliquer dans les parties précédentes, si l'apprentissage profond peut conduire à de très bonnes performances, les transformations opérées restent non interprétables dans une large mesure. De plus, le réhaussement de la parole suscite également un intérêt particulier pour les algorithmes à faible coût de calcul. En effet, certaines des principales applications dans ce domaine nécessitent un traitement en temps réel, comme l'illustre le challenge Deep Noise Suppression (DNS) [62] où les modèles doivent être exécutés en temps contraint sur des ressources matérielles précises.

7.1.1 Motivations

Les architectures typiques des réseaux neuronaux exploitent des filtres internes pour traiter l'information. Toutefois, les filtres utilisés dans chaque couche sont choisis de manière ad hoc et n'ont qu'un rapport vague avec la nature du signal traité. Cependant, comme on a pu le décrire dans le chapitre précédent, on peut contraindre la structure des convolutions de la première couche du modèle afin d'améliorer l'interprétabilité des modèles tout en conservant (voire en augmentant) les performances pour des tâches de séparation de sources.
Dans ce chapitre, nous proposons une approche pour apprendre des filtres interprétables au sein d'une architecture de réseau de neurones spécifique qui permet de mieux comprendre le comportement du réseau et de réduire sa complexité à tous les étages du modèle. L'architecture choisie pour cette étude est une architecture traitant le signal dans le domaine de la forme d'onde, le Wav-UNet [35] : cette architecture s'est montrée efficace dans les applications de séparation de sources, et, par sa structure simple, elle se montre également flexible pour tendre vers un cadre plus interprétable.

7.1.2 Wave-UNet : principe et architecture

Nous sommes particulièrement intéressés ici par les architectures de type Wav-UNet, qui traitent la forme d'onde directement dans le domaine temporel. L'intérêt d'étudier cette architecture réside dans la flexibilité de ces composants puisqu'il s'agit uniquement d'une succession de convolutions 1D (temporelles) utilisant des pas d'avancement de 1 combinées à des fonctions de sous-échantillonnage ou d'interpolation. Cette architecture a prouvé son efficacité pour des tâches de séparation de sources et de rehaussement de la parole lorsqu'elle est utilisée seule ou lorsqu'elle est couplée à une seconde branche avec une représentation temps/fréquence comme dans [96]. Un autre aspect intéressant de cette architecture est la possibilité d'adapter la résolution temporelle (via la quantité de sous-échantillonnage utilisée) mais aussi la résolution en fréquence (via la taille des filtres utilisés dans les convolutions). Nous rappelons brièvement ci-dessous l'architecture du modèle Wav-UNet.

Architecture globale de Wav-UNet

Wav-UNet est une architecture de type encodeur-décodeur bout-en-bout qui opère directement sur la forme d'onde audio brute. À chaque couche de l'encodeur, la résolution temporelle est réduite à l'aide d'une cascade de convolutions 1D suivies d'opération de sous-échantillonnage (décimation d'un facteur 2 dans l'article original). Au niveau du décodeur, le signal est reconstruit en empilant également des convolutions 1D. Ici, le sur-échantillonnage est obtenu par une interpolation linéaire des valeurs. L'entrée de chaque couche de décodage est la sortie de la couche de décodage précédente concaténée avec la couche d'encodage symétrique. Cette architecture est résumée dans la Fig. 7.1 où T représente la durée (en échantillons) du signal d'entrée, $C_l \in \{1 \dots L\}$ le nombre de filtres de la couche l (qui est le même dans l'encodeur et le décodeur).
Chaque bloc de convolution dans l'encodeur consiste en une convolution 1D suivie d'une opération de sous-échantillonnage (DS), d'une batch-normalisation (BN) et d'une ReLU :

$$\mathbf{E_l} = \mathrm{DS}(\mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}_l(\mathbf{E}_{l-1})))), \tag{7.1}$$

$\mathbf{E}_l \in \mathbf{R}^{C_l \times T/(2^l)}$ représente la projection du signal dans la $l^{\text{ième}}$ couche.
Le décodeur est l'exact opposé de l'encodeur. Le sur-échantillonnage (US) est effectué à l'aide d'une interpolation linéaire entre les valeurs successives.

$$\mathbf{D_{l-1}} = \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}_l(\mathrm{US}([\mathbf{D}_l) : \mathbf{E}_l)]))), \tag{7.2}$$

$[\mathbf{D}_l : \mathbf{E}_l] \in \mathbf{R}^{(2C_l) \times T/(2^l)}$ désigne la concaténation de la projection décodée précédente et de la projection encodée symétriquement.
La sortie du décodeur $\mathbf{D}_1$ est une carte de caractéristiques à la même résolution temporelle que la forme d'onde audio d'entrée. Une dernière couche de convolution est utilisée sur la concaténation de $\mathbf{D}_1$ et du mélange $\mathbf{x}$ pour reconstruire le signal propre $\hat{\mathbf{v}}$.

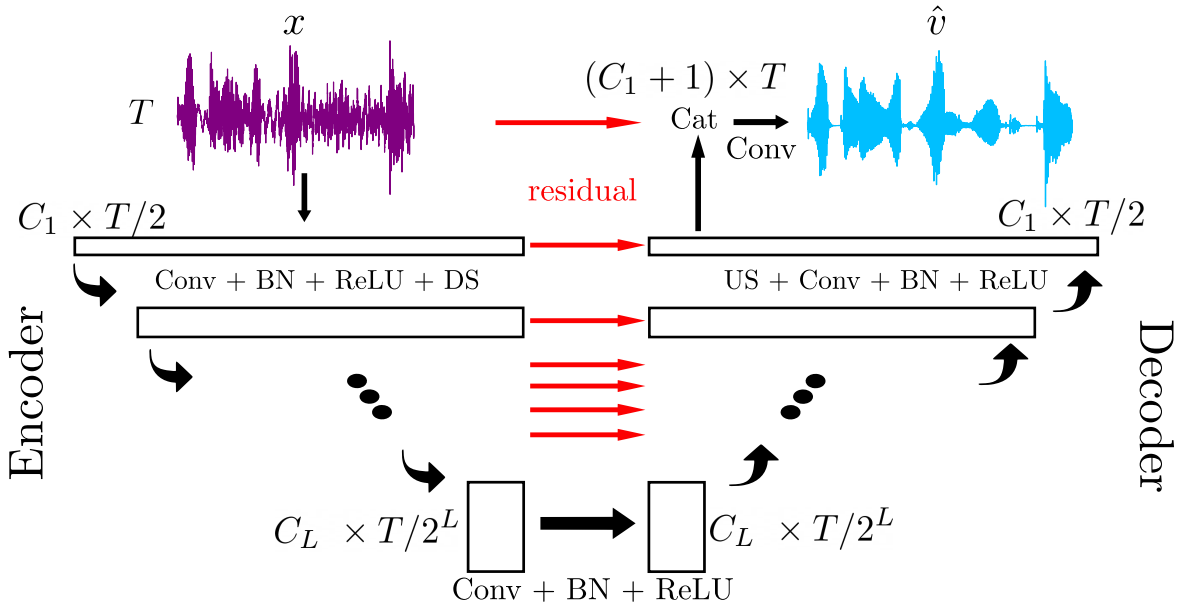


FIGURE 7.1 – Modèle WavUNet. Les flèches noires indiquent les blocs de convolutions. Les flèches rouges indiquent les connexions résiduelles (concaténation dans ce modèle). $T_l \times C_l$ désigne la dimension de la projection à la couche l.

7.1.3 Propositions

L'intérêt de cette architecture réside dans le fait que l'on travaille le signal dans le domaine temporel, ainsi, on peut essayer de conserver une structure de signal à tous les niveaux de projections du modèle.

Modification de la résolution temporelle et fréquentielle

Une simple modification de la structure de cette architecture permet de modifier simultanément les résolutions temporelle et fréquentielle au sein du réseau. En effet, la taille K des filtres définit la résolution en fréquence tandis que l'échantillonnage utilisé à chaque couche définit la résolution temporelle (obtenue ici en modifiant le facteur de décimation).
Notons cependant que l'utilisation de filtres de plus grande taille augmente significativement le coût de calcul. Nous proposerons ci-dessous une stratégie qui permettra d'exploiter des filtres de plus grande taille sans augmenter le coût de calcul et en conservant l'expressivité du modèle (c'est-à-dire sans affecter le nombre de canaux à travers le réseau).

Modification des non-linéarités

Une autre piste intéressante à appliquer sur cette architecture est la modification des fonctions de non-linéarités. En effet, l'utilisation de non-linéarités ReLU entraîne de lourds artefacts dans la décomposition fréquentielle du signal. On cherchera ici

d'autres types de non-linéarité permettant potentiellement de conserver au maximum des structures fréquentielles claires à tous les étages du modèle.

7.2 Apprentissage de filtres interprétables

L'objectif de cette section est double :
— introduire des filtres interprétables dans l'architecture Wav-Unet (ce qui permettra une meilleure visualisation des étapes intermédiaires du réseau),
— permettre une réduction importante du coût de calcul et réduire le volume de données d'apprentissage.

À cette fin, nous proposons de paramétrer les filtres de convolution de manière à ce que les représentations cachées des différentes couches restent interprétables en termes de contenu temps-fréquence à différentes résolutions. Pour le second objectif, nous proposons de factoriser les filtres paramétrés et d'exploiter le concept de convolutions séparables [97, 98].

7.2.1 Convolutions séparables dans Wav-UNet

La convolution séparable [97] est une convolution dont le noyau est factorisé en deux termes qui permet de réduire le coût de calcul global [98, 99] d'une couche de convolution. Cependant, les convolutions séparables ne permettent de représenter qu'un sous-ensemble de toutes les convolutions, elles sont donc potentiellement moins expressives que des convolutions classiques. La convolution séparable factorise les filtres de convolutions en une succession de deux opérations indépendantes :
— Une convolution *depthwise* C_d : qui convolue chaque canal des caractéristiques d'entrée indépendamment avec un filtre de taille dans l'axe temporel K et de profondeur 1 dans l'axe des canaux,
— Une convolution *pointwise* C_p : qui correspond à une couche linéaire de C_l canaux vers C_{l+1} canaux appliquée indépendamment à chaque pas de temps.

La sortie d'une telle convolution est exprimée comme suit :

$$\text{Conv}_{DS}X = C_p(C_d(X)), \tag{7.3}$$

Le gain de complexité apporté par cette convolution dans le Wav-UNet (voir la partie 7.4.5 pour plus de détails) permet d'utiliser des fenêtres de convolutions plus grandes, à coût de calcul constant. Cet agrandissement permet aux filtres d'avoir une résolution fréquentielle plus élevée, et donc des interprétations plus précises de leurs caractéristiques.

À la lumière de la remarque ci-dessus, nous pouvons donc paramétrer et régler explicitement tous les filtres pour toutes les couches, comme on a pu l'étudier dans le chapitre précédent 6. Cela nous permet d'obtenir des filtres ayant des réponses fréquentielles plus éparses en minimisant les composantes de bruit dans les domaines temporel et fréquentiel tout en réduisant le nombre de paramètres.

7.2.2 Convolutions depthwise paramétrées

Notre proposition consiste à utiliser la partie réelle de filtres paramétrés de Gabor 6.3.1 pour toutes les couches du réseau :

$$g(n|\nu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{n^2}{\sigma^2}} \cdot e^{j2\pi\nu n}, \tag{7.4}$$

où ν et σ sont des paramètres entraînables représentant la fréquence centrale normalisée et la largeur de bande du filtre. Les filtres de Gabor complexes ont été envisagés dans un premier temps (car ils peuvent améliorer l'invariance du déphasage du réseau), mais n'ont finalement pas été retenus pour cette étude car nous utilisons ici de petites valeurs de pas d'avancement (un ou deux échantillons). Contrairement au chapitre précédent, nous n'avons pas besoin d'avoir des filtres ayant une capacité de localisation temporelle ou une sur-paramétrisation de phase.

7.3 Non-linéarité

L'utilisation de filtres de Gabor permet effectivement d'obtenir des représentations passe-bande du signal en sortie des couches de convolutions. Cependant, l'utilisation d'une non-linéarité ReLU perturbe la structure fréquentielle de ces représentations, l'objectif de cette section est de proposer d'autres non-linéarités permettant de conserver convenablement la structure fréquentielle des représentations latentes.

7.3.1 Cas de la ReLU

L'application d'une non-linéarité ReLU sur un signal réel aura pour effet d'ajouter des composantes fréquentielles dans tout le spectre. En effet, si l'on prend une sinusoïde de fréquence ν_0, l'application d'une ReLU sur celle-ci, est équivalent dans le domaine fréquentiel à convoluer $\mathcal{F}(\sin(\nu_0 t))$ par un peigne de Dirac de fréquence ν_0. On illustre ce phénomène dans la figure 7.2.
De plus, la réponse fréquentielle de la ReLU sera d'autant plus complexe à mesure que le signal considéré aura un support fréquentiel fourni. La représentation temporelle en créneau de la fonction ReLU aura toujours pour effet d'étaler le spectre du signal d'origine.
On cherche donc à construire une non-linéarité temporelle qui minimise les distorsions fréquentielles du signal afin d'assurer la représentation du signal à être passe-bande.

7.3.2 CLU : Compressor Linear-Unit

Idéalement, on voudrait obtenir une non-linéarité symétrique par rapport à l'axe d'amplitude afin de conserver la structure des signaux. L'utilisation de la tanh est exclue, car elle ne permet pas de rétropropager convenablement les gradients.

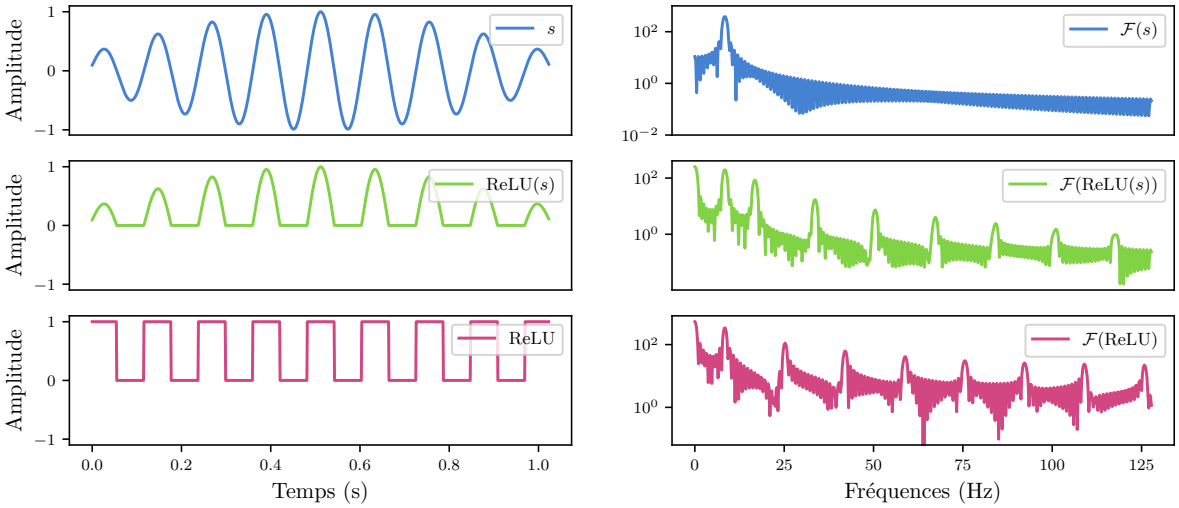


FIGURE 7.2 – Exemple d'application d'une ReLU sur un signal simple.

On propose ici de définir une non-linéarité composée de fonction affine par morceau comme cela a été fait par Zhu et al. [14], mais dans notre cas, avec une contrainte d'imparité sur la non-linéarité :

$$\text{CLU}(x|\tau, c) = \begin{cases} x & \text{si } |x| \leq \tau, \\ \frac{x - \text{sign}(x) * \tau}{c} + \text{sign}(x)\tau & \text{sinon,} \end{cases} \tag{7.5}$$

où τ correspond à un seuil et où c correspond à un facteur de compression. Ces paramètres peuvent être optimisés durant l'apprentissage du modèle. La figure 7.3 présente un exemple d'une telle non-linéarité.

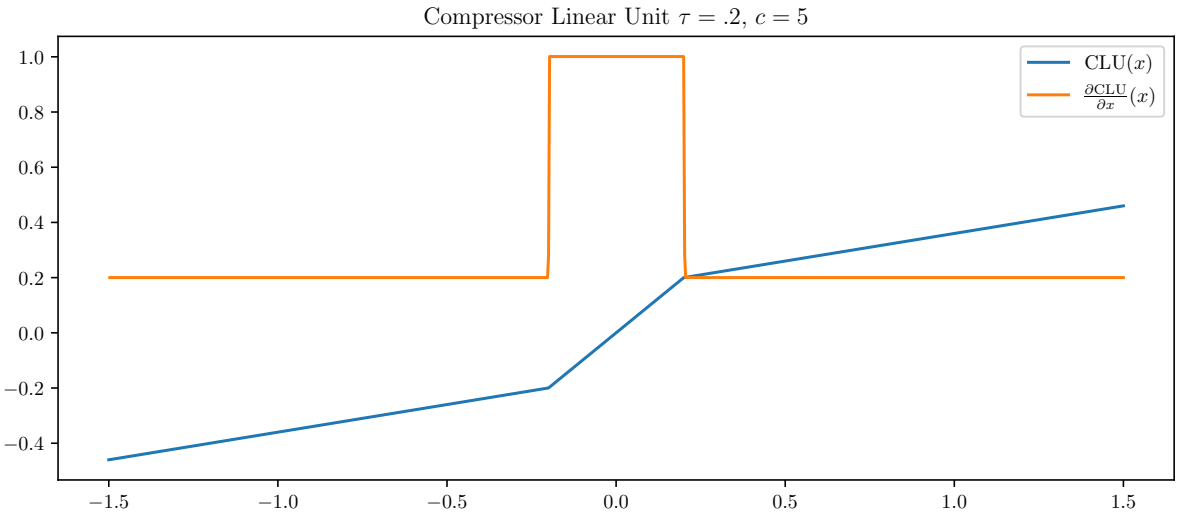


FIGURE 7.3 – Non-linéarité CLU et sa dérivée

Utilisée telle quelle, cette non-linéarité ne fait apparaître que de faibles distorsions dans le domaine fréquentiel. Cependant, les distorsions temporelles associées peuvent également être très faibles. Nous proposons donc également de joindre en sortie de la

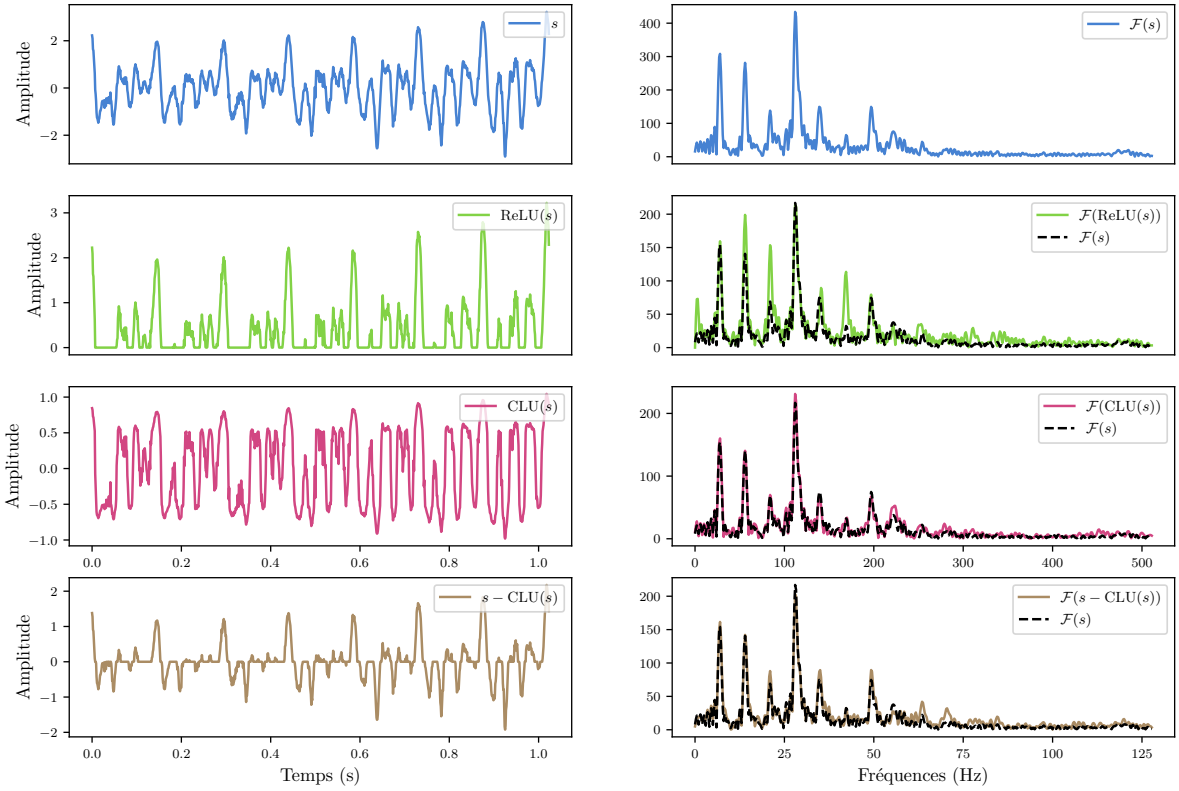


FIGURE 7.4 – Réponse temporelle et fréquentielle induite par CLU et ReLU sur un signal de parole.

non-linéarité la différence entre le signal d'entrée et sa version compressée $s - \text{CLU}(s)$, qui, comme on peut le voir dans la figure 7.4, n'apporte que peu de distorsion dans le domaine fréquentiel.
Le nombre de canaux en sortie de la non-linéarité est donc multiplié par 2. Afin de comparer les versions des modèles utilisant ou pas cette non-linéarité, on adaptera les hyper-paramètres du modèle afin d'avoir un nombre de FLOPS constants.

7.3.3 Contrainte de biais

Notons que l'utilisation de non-linéarité CLU dans un Wav-UNet peut amener à une correspondance avec la non-linéarité PReLU [68] :

$$\text{PReLU}(x|\alpha) = \begin{cases} x & \text{si } x \geq 0, \\ \alpha x & \text{sinon,} \end{cases} \tag{7.6}$$

si les biais appris par les différentes couches de convolutions et de batch normalisations décale le signal de manière trop importante. Avant d'appliquer cette non-linéarité, il faut donc recentrer les différents canaux pour minimiser les distorsions fréquentielles.

7.4 Expériences

Dans cette section, nous décrivons la mise en œuvre expérimentale des propositions développées plus haut, analysons les propriétés des filtres appris et présentons nos résultats sur la tâche spécifique du rehaussement de la parole.

7.4.1 Paramètres expérimentaux

Base de données

Nous utilisons l'ensemble de données du challenge DNS [67]. Les exemples sont créés à la volée à partir de différentes voix, d'arrière-plans et de réponses impulsionnelles avec un SNR compris entre -5 et +20 dB. Les ensembles d'entraînement, de validation et de test sont construits de manière à ce que les locuteurs et les arrière-plans ne soient pas identiques entre les différents ensembles.

Caractéristiques des modèles

Tous les modèles sont des Wav-UNet à 9 couches avec $24 \times l$ canaux à la couche l. Le facteur de décimation est fixé à 2. La dernière couche de l'encodeur produit donc une représentation cachée de taille $216 \times T/2^9$, où T est la taille initiale du signal d'entrée. Pour le modèle de base, la taille du noyau est fixée à 15. Pour les autres modèles, les convolutions séparables sont paramétrées de manière à avoir autant de canaux que le modèle de base, mais avec une taille de filtre K fixée à 64. La dernière couche de l'encodeur produit donc une représentation cachée de taille $216 \times T/2^9$, où T est la taille initiale du signal d'entrée.

Protocole expérimental

Dans ce chapitre, nous comparons trois architectures différentes : le modèle de base, un modèle à encodeur séparable (SE-WavUNet) où le décodeur utilise des convolutions standard et un modèle entièrement séparable (FS-WavUNet) où l'encodeur et le décodeur utilisent des convolutions séparables. Les filtres des convolutions séparables des modèles SE et FS sont testés en paramétrage libre et en paramétrage de Gabor (Gabor-WavUNet). L'entraînement de tous les modèles est réalisé avec des mélanges de 3 secondes. On utilise le Si-SNR comme fonction de coût permettant l'optimisation du modèle avec l'algorithme d'optimisation Adam et un taux d'apprentissage de 10^{-3}. Les performances sont mesurées à l'aide du SDR, du STOI [60] et du PESQ [59] afin de comparer les modèles.

7.4.2 Propriétés des filtres appris

Visualisation des Wav-UNets *Fully-Separable*

Nous étudions ici les propriétés des filtres appris (avec une taille de noyau $K = 64$). Nous illustrons dans la Fig 7.5 l'amplitude de la transformée de Fourier discrète (DFT) des filtres de résolution 1D appris pour la première et la quatrième couche dans le codeur modèle. Comme on peut le voir, les filtres sont bien localisés dans le domaine des fréquences.

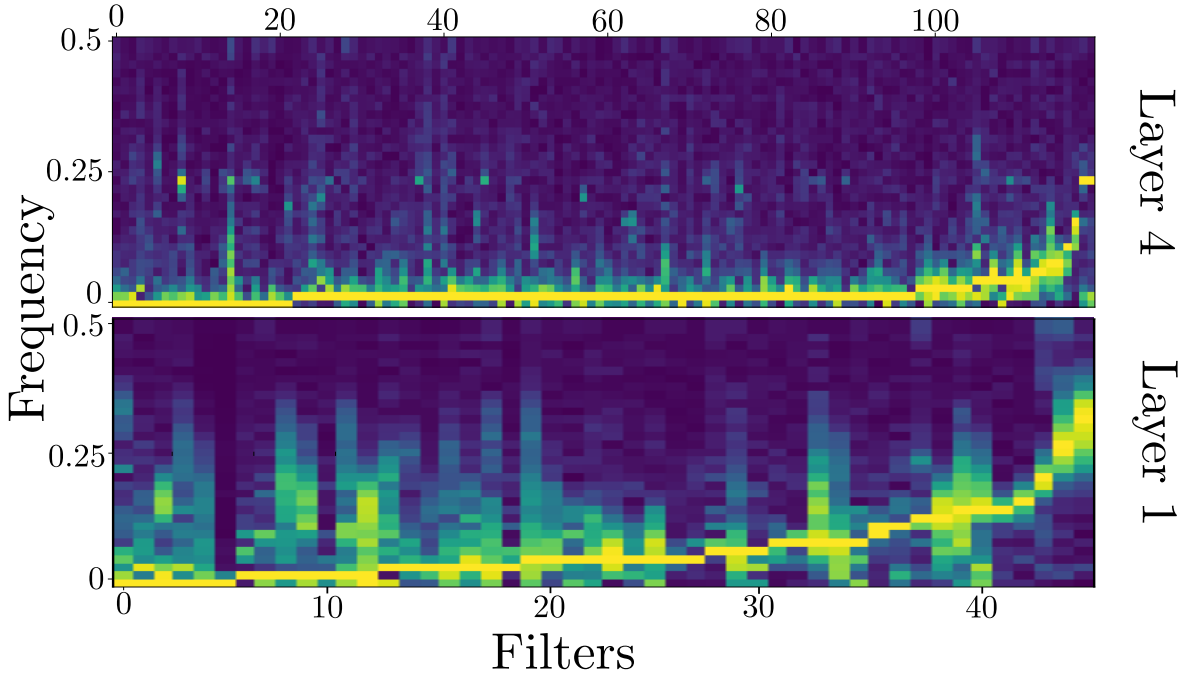


FIGURE 7.5 – Amplitude de la TFD des filtres de convolution 1D utilisés pour les couches $l = 1$ et $l = 4$ dans la partie encodeur d'un FS-WavUNet.

Ce comportement s'étend à toutes les couches autant dans la partie encodeur que décodeur du modèle. La faible résolution temporelle semble entraîner l'apprentissage d'une large gamme de filtres passe-bas. Les Fig. 7.6 et Fig. 7.7 présentent en outre les fréquences centrales normalisées (définies ici comme l'arg max de la TFD des valeurs du filtre) de tous les filtres pour toutes les couches pour un même réglage.
À partir de là, nous pouvons également noter que, dans la grande majorité des cas, le système a appris des filtres avec une fréquence centrale normalisée $\nu \leq 0,25$. Cette propriété est très intéressante car elle indique qu'il est possible d'utiliser un pas d'avancement de 2 dans les convolutions sans perdre d'information.
En effet, un pas d'avancement de 2 ou une décimation d'un facteur 2 sont strictement équivalents pour l'encodeur dans ce scénario. Ce comportement réduit de moitié le coût de calcul dans l'encodeur sans aucune modification dans la prédiction finale.

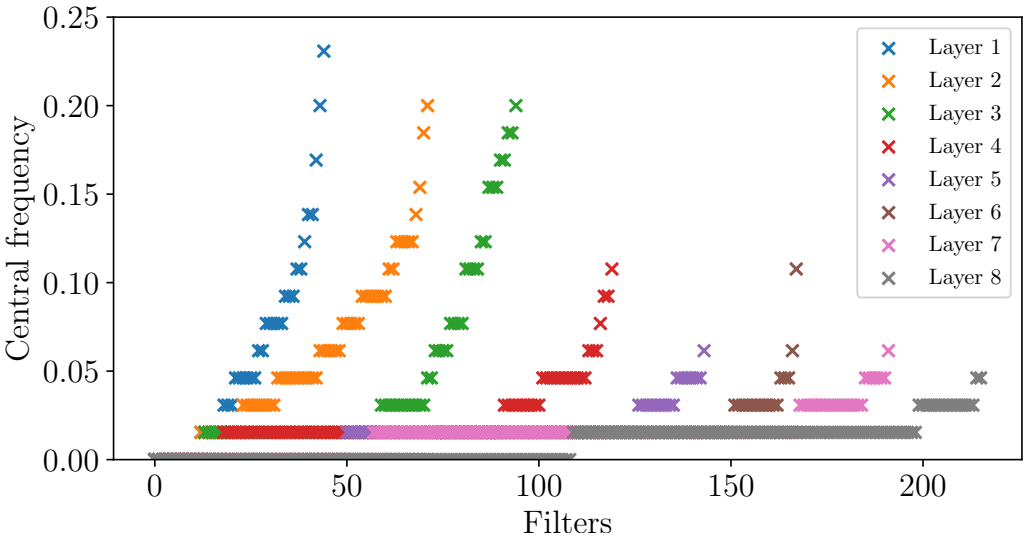


FIGURE 7.6 – Fréquences centrales normalisées des filtres appris pour chaque couche d'un FS-WavUNet.

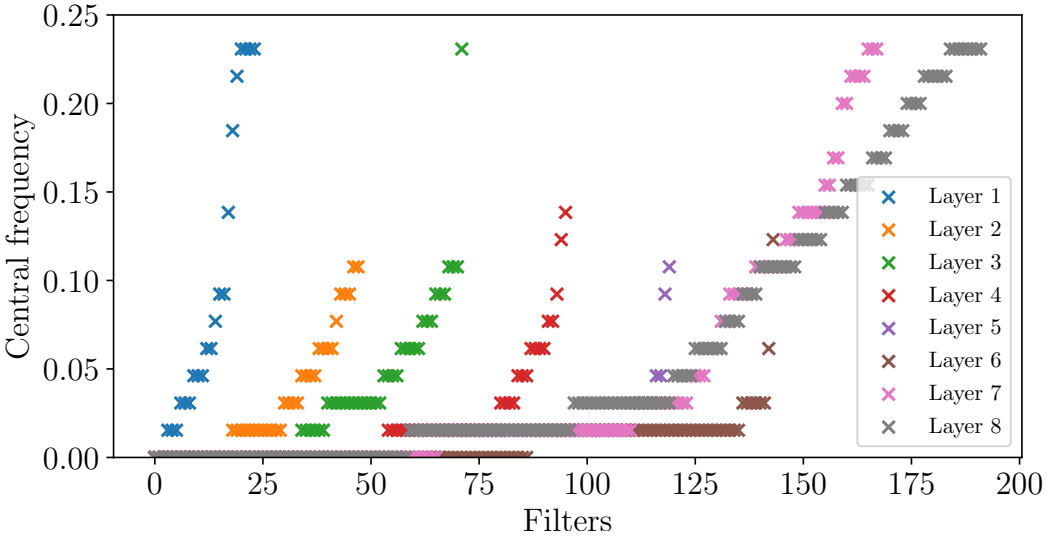


FIGURE 7.7 – Fréquences centrales normalisées des filtres appris dans le décodeur (FS-WavUNet).

Visualisation du modèle Gabor-WavUNet

Les analyses précédentes suggèrent d'utiliser des filtres de Gabor à toutes les couches. En effet, ce paramétrage nous permet de construire des filtres dont la fréquence centrale et la largeur de bande sont contrôlables. De plus, nous pouvons également limiter la fréquence centrale à prendre des valeurs inférieures à 0,25 afin d'utiliser un pas d'avancement de 2 sans perte de généralité. Dans la Figure 7.8, nous illustrons les valeurs de

ν pour tous les filtres de Gabor (de taille 64) de toutes les couches.

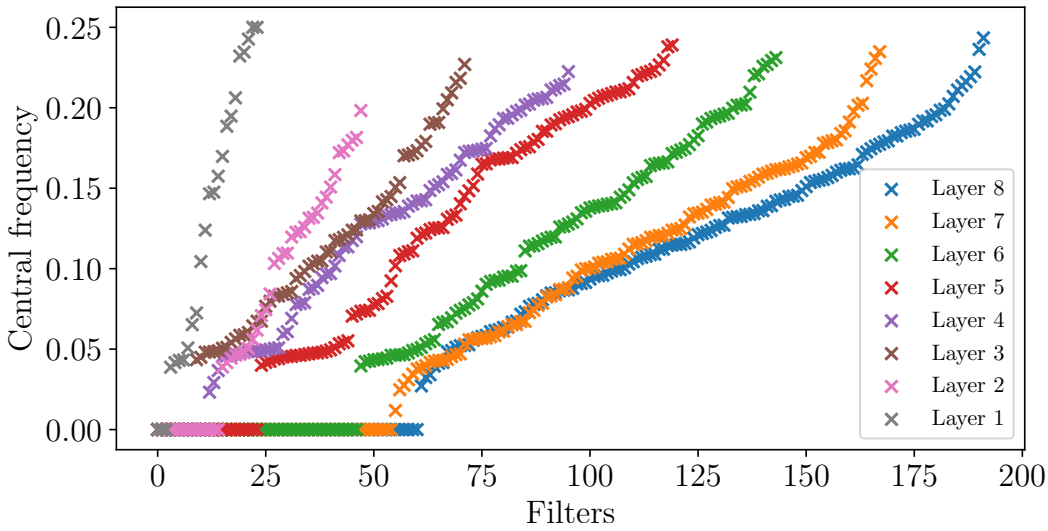


FIGURE 7.8 – Fréquences centrales des filtres appris pour chaque couche d'un Gabor-WavUNet dans l'encodeur.

Les figures 7.6 et 7.8 présentent le comportement des filtres appris dans les premières couches. La principale différence entre les filtres de Gabor et les filtres classiques se situe au niveau des fréquences centrales des dernières couches. Bien que clairement perceptible, nous n'avons pas d'interprétation solide de cette différence pour les dernières couches. Nous notons également, pour toutes les couches, une discontinuité près des fréquences zéro. Pour l'expliquer, nous étudions les largeurs de bande associées à ces filtres, et, comme l'illustre la Figure 7.9 pour la couche 8, les filtres ont des largeurs de bande importantes, ce qui explique qu'ils peuvent couvrir les basses fréquences "manquantes".

7.4.3 Résultats sur le rehaussement de la parole

Nous donnons dans le tableau 7.1 les performances des différents modèles sur la tâche de rehaussement de la parole : *Noisy* indique le niveau moyen de détérioration dans l'ensemble de données tandis que les modèles SE utilisent des convolutions séparables dans l'encodeur et FS sont entièrement séparables. On constate que tous les modèles obtiennent les mêmes résultats pour toutes les mesures. Deux points méritent d'être soulignés :

— premièrement, l'utilisation de convolutions séparables maintient des performances équivalentes tout en réduisant les coûts de calcul ;
— deuxièmement, la paramétrisation avec des filtres de Gabor donne des résultats similaires.

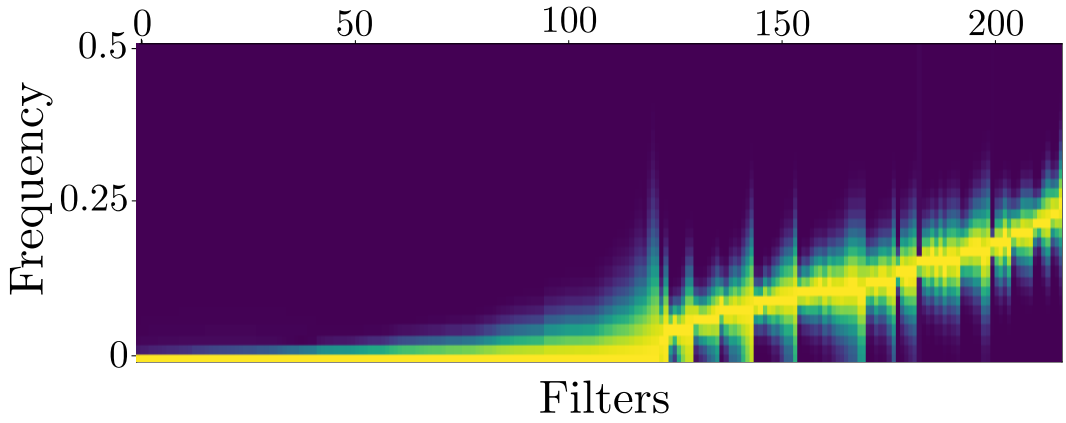


FIGURE 7.9 – Fréquences centrales des filtres appris dans un modèle Gabor-WavUnet. $l = 8$.

Ainsi, le nombre de paramètres dans les filtres des convolutions séparables peut considérablement être réduit (notons néanmoins qu'une grande partie des paramètres se trouve dans la partie *pointwise*, qui n'est pas étudiée dans le présent chapitre). En outre, l'utilisation de filtres de Gabor permet d'obtenir à la sortie de chaque couche des signaux passe-bande bien localisés en temps et en fréquence. L'exploitation de cette propriété en vue d'améliorer encore la méthode est toutefois laissée pour des travaux futurs. Dans ce chapitre, nous n'essayons pas de tirer parti de cette structure, mais il s'agit d'une base intéressante pour des études ultérieures.

TABLE 7.1 – Performance des différents modèles. L'écriture $x \pm y$ correspond à une valeur moyenne x et un écart type y sur le jeu de données. L'écart type y s'explique essentiellement par la dispersion des valeurs de SNR dans l'ensemble de données de test. Les modèles SE correspondent aux Wav-UNet où seul l'encodeur est composé de convolutions séparables et les modèles FS où l'encodeur et le décodeur sont composés de convolutions séparables.

	SNR	STOI	PESQ
Noisy	7.33 ± 3.45	0.84 ± 0.12	2.15 ± 0.70
BaseLine	14.41 ± 2.42	0.88 ± 0.10	2.85 ± 0.57
SE	14.36 ± 2.68	0.88 ± 0.10	2.86 ± 0.60
Gabor-SE	14.49 ± 2.54	0.88 ± 0.12	2.83 ± 0.60
FS	14.36 ± 2.67	0.89 ± 0.10	2.88 ± 0.59
Gabor-FS	14.15 ± 2.48	0.89 ± 0.09	2.83 ± 0.57

7.4.4 Résultats sur l'utilisation de CLU

Dans le tableau 7.2, on peut voir que l'utilisation de la non-linéarité CLU avec batch normalisation permet d'obtenir des performances comparables à celles obtenues

avec l'utilisation d'une non-linéarité ReLU. Cependant, et comme expliqué dans la partie 7.3.3, en laissant la fonction de batch-normalisation telle quelle, les biais appris par le modèle rendent la fonction CLU semblable à celle d'une fonction PReLU. En enlevant le biais de la non-linéarité, on diminue sensiblement les performances du modèle. Cependant, les représentations latentes ainsi obtenues sont des signaux passe-bande très bien définis en temps et en fréquences.
Même si l'utilisation propre de CLU diminue les performances du WavUNet, on peut quand même noter que l'obtention de signaux très bien définis en temps et en fréquences à tous les niveaux du modèle peuvent être à la base d'extensions expertes.

TABLE 7.2 – Comparaison des performances avec et sans utilisations de CLU. Le modèle Gabor-FS-CLU-BN correspond à un modèle appris avec la non-linéarité où la batch-normalisation est laissée telle quelle. Le modèle Gabor-FS-CLU-noBN correspond à une batch-normalisation dans laquelle le paramètre de biais est supprimé. L'écriture $x \pm y$ correspond à une valeur moyenne x et un écart type y sur le jeu de données. L'écart type y s'explique essentiellement par la dispersion des valeurs de SNR dans l'ensemble de données de test.

	SNR	STOI	PESQ
Noisy	7.33 ± 3.45	0.84 ± 0.12	2.15 ± 0.70
Gabor-FS	14.15 ± 2.48	0.89 ± 0.09	2.83 ± 0.57
Gabor-FS-CLU-BN	14.31 ± 2.57	0.89 ± 0.11	2.82 ± 0.59
Gabor-FS-CLU-noBN	12.44 ± 2.87	0.86 ± 0.11	2.69 ± 0.63

7.4.5 Analyse de la complexité

Nous comparons brièvement dans le tableau 7.3 le nombre de paramètres à apprendre et le nombre d'opérations en virgule flottante par seconde (FLOPS) nécessaires pour traiter une seconde de signal. L'utilisation de convolutions séparables permet de réduire la complexité (en termes de FLOPS et de nombre de paramètres) de l'architecture Wav-UNet tout en conservant les mêmes niveaux de performance (voir tableau 7.1). Notons que le nombre de paramètres libres est plus drastiquement réduit dans cette architecture que dans l'apprentissage de front-ends contraints ce qui devrait permettre d'utiliser moins de données d'apprentissage. Cette propriété a priori est partiellement vérifiée en pratique comme le montre le tableau 7.4. L'approche semble apporter un gain de performance pour une base de données d'apprentissage 10 fois plus petite que la base complète. Pour d'autres volumes de données, les performances du modèle paramétrique et du modèle complètement appris sont similaires.
Notons comme défaut que même si le nombre de FLOPS théoriques est réduit entre les modèles, les calculs de convolutions séparables sont moins bien parallélisés que les filtres classiques sur pyTorch et encore plus avec des filtres de grande taille. Le temps

TABLE 7.3 – Flops et nombre de paramètres pour chaque modèle discuté dans cet article.

Modèle	# Paramètres	FLOPS
BaseLine	3.8M	1.66G
SE	2.70M	1.38G
Gabor-SE	1.43M	1.38G
FS	1.1M	0.42G
Gabor-FS	0.76M	0.42G

TABLE 7.4 – Performance (SiSNR) en fonction de la quantité de données utilisées durant l'apprentissage pour les modèles FS et Gabor-FS. L'écriture $x \pm y$ correspond à une valeur moyenne x et un écart type y sur le jeu de données. L'écart type y s'explique essentiellement par la dispersion des valeurs de SNR dans l'ensemble de données de test.

Modèle / Data	0.1%	1%	10%	100%
FS	11.86 ± 2.82	12.42 ± 2.42	12.81 ± 2.49	14.36 ± 2.67
Gabor-FS	11.94 ± 2.94	12.66 ± 2.49	13.31 ± 2.31	14.15 ± 2.48

d'apprentissage et d'inférence est donc assez similaire avec l'implémentation actuelle de pyTorch.

7.5 Conclusion

Dans ce chapitre, nous avons proposé une méthode d'apprentissage de filtres interprétables pour toutes les couches d'une architecture neuronale dédiée (Wav-Unet). Nous avons montré comment ces filtres sont interprétables et comment nous avons exploité leurs propriétés pour introduire un gain de complexité significatif. Nous avons également démontré que les gains en termes d'interprétabilité et de complexité n'ont pas d'impact négatif sur la performance du modèle dans une tâche de réhaussement de la parole. De plus, nous avons également proposé d'utiliser un nouveau type de non-linéarité pour satisfaire la propriété de symétrie des signaux audio. Même si l'utilisation de cette non-linéarité entraîne une baisse des performances, les représentations induites par ce modèle sont d'autant plus interprétables en tant que signaux audio passe-bande. De futurs travaux pourront être consacrés à l'extension de la méthode à d'autres applications audio (par exemple, la séparation de sources musicales, la détection de scènes acoustiques et d'événements) afin d'évaluer la généricité des filtres obtenus et de concevoir, si nécessaire, des stratégies d'adaptation appropriées.

Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés à différentes méthodes pour mieux modéliser l'information de phase dans les réseaux de neurones profonds. L'objectif était en même temps de tester différentes manières de représenter la phase dans les réseaux de neurones (chapitre 5 et chapitre 6) mais également de construire des méthodes permettant d'éviter de passer par des représentations nécessitant la modélisation la phase (chapitre 7).

Rappel des contributions

Utilisation des réseaux de neurones à valeurs complexes

Dans le chapitre 5 nous nous sommes intéressés à l'utilisation des réseaux de neurones à valeurs complexes pour modéliser d'une part la phase de la transformée de Fourier court terme et d'autre part pour analyser le signal de manière bout-en-bout dans le domaine complexe. On a montré que dans l'état actuel des choses, les architectures complexes ne permettaient pas d'apporter, ni une augmentation des performances par rapport à un modèle réel, ni d'ajouter de l'interprétabilité sur le traitement de la phase dans le modèle. On a également pu montrer, que c'est l'utilisation de canaux codépendants des parties réelles et imaginaires qui aide le modèle à prendre en compte la phase. Ce premier chapitre de contributions nous a permis de nous focaliser sur les réseaux de neurones à valeurs réels dans la suite des travaux.

Bancs de filtres pour la séparation de sources

Dans le chapitre 6 nous nous sommes intéressés à la construction de bancs de filtre réels afin de modéliser l'information de phase des signaux de manière implicite dans un modèle réel. Nous avons déterminé un certain nombre de propriétés souhaitables pour dans le banc de filtres du frontend pour aider à la séparation pour des modèles de type TasNet. Plus particulièrement, on a mis en évidence des critères pour s'assurer d'obtenir des invariances aux décalages temporels et aux décalages de phases. De plus, on a montré que l'utilisation de bancs de filtres sur-paramétrés en phase permettait d'obtenir de meilleures performances tout en diminuant le nombre de paramètres. Enfin, on a présenté un banc de filtres paramétrés (banc de filtres de Bedrosian) qui permet de garantir de bonnes propriétés (invariance à la phase et aux décalages temporels) tout en étant suffisamment flexible pour modéliser des filtres de nature très différente.

Extension des bancs de filtres à l'ensemble d'un modèle

Dans le chapitre 7, on s'est intéressé à l'extension de l'utilisation des bancs de filtres à l'ensemble des convolutions d'un modèle. Pour cela, on a adapté les convolutions d'un modèle de type WavUNet afin de pouvoir y intégrer des bancs de filtres paramétrables. En particulier, on a proposé d'utiliser des filtres de Gabor afin d'optimiser au mieux les résolutions en temps et en fréquence des représentations issues du Wav-UNet. De cette manière, on s'assure d'obtenir des représentations passe-bandes du signal dans toutes les couches du modèle. De plus, cette paramétrisation permet de diminuer le coût computationnel et le nombre de paramètres d'un Wav-UNet standard tout en permettant de conserver les performances dans une tâche de rehaussement de la parole. Enfin, on a essayé de remplacer les non-linéarités ReLU du modèle afin d'assurer le caractère symétrique (et oscillant) des représentations internes. Cependant, cette tentative ne permet pas de conserver les performances d'un modèle utilisant des non-linéarités ReLU.

Perspective

Les pistes présentées dans ce document sont loin d'être exhaustives quant à la manière de modéliser la phase dans les réseaux de neurones. La modélisation de la phase est une problématique complexe et l'on s'est essentiellement concentré sur des méthodes de modélisation interne aux réseaux de neurones.

Fonction de coût invariante à la phase

Dans les travaux menés ici, les fonctions de coût utilisées étaient essentiellement des fonctions de coût dépendantes à la phase. C'est-à-dire qu'une production d'un signal parfait pour l'oreille humaine mais ayant une phase différente serait tout de même pénalisée par des fonctions de coût comme le Si-SNR ou des fonctions de coût spectrales complexes. Cependant, si le signal à reconstruire est fortement distordu, il n'est pas évident que la phase de certaines composantes fréquentielles soit possible à déterminer. Il serait donc judicieux de trouver des critères d'invariances à la phase dans les fonctions de coût utilisées. Les fonctions de coût sur l'amplitude du spectrogramme donnent un exemple de telle fonction de coût invariante à la phase. Comme piste de réflexion, on pourrait, par exemple, étendre cette fonction de coût en prenant un autre banc de filtres que celui de la TFCT (comme des filtres de Bedrosian) et où le pas d'avancement pourrait être également appris comme cela été fait par Riad et al. [100].

Non-linéarités

L'utilisation de la non-linéarité ReLU n'est pas satisfaisante pour faire circuler l'information de phase dans les réseaux de neurones.

En effet, comme on a pu en discuter dans le chapitre 5 dans le cas des réseaux de neurones à valeurs complexes, l'utilisation de la ReLU entraîne une perte de l'information de phase et rend la pertinence des RNVC caduques.
De plus, dans le chapitre 7 on a pu voir que la non-linéarité ReLU distord fortement les composantes fréquentielles des signaux en faisant apparaître des harmoniques des fréquences actives. Celles-ci peuvent parasiter la qualité de séparation dans ce type d'approche. L'utilisation de la non-linéarité CLU ne s'est pas montré pertinente pour améliorer les performances, cependant elle offre des représentations internes plus proches de la nature des signaux audio.
La poursuite des recherches autour des non-linéarités semble donc être une piste particulièrement intéressante pour traiter convenablement la phase des signaux dans les réseaux de neurones.

Extension des travaux à d'autres tâches que la séparation de sources

Dans le chapitre 6, les bancs de filtres construit ont été testés exclusivement sur des tâches de séparation de sources. Cependant, les bancs de filtres ont montré être capables d'affecter les performances dans d'autres tâches d'analyse comme la reconnaissance de locuteur [77]. On pourrait donc logiquement appliquer les travaux effectués dans ce chapitre sur de nouvelles tâches. En particulier, on pourrait appliquer ces bancs de filtres sur des tâches d'analyse comme la reconnaissance de locuteur, mais également sur d'autres tâches d'analyse-synthèse que la séparation de sources comme la compression.

Bibliographie

[1] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice-hall international editions edition, 1975.

[2] C.E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, January 1949.

[3] L.R. Rabiner and R.W. Schafer. *Digital Processing of Speech Signals*. Prentice-hall international editions edition, 1978.

[4] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. January 1999.

[5] S.S. Stevens, J. Volkmann, and E.B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch | The Journal of the Acoustical Society of America | AIP Publishing, 1937.

[6] Judith Brown. Calculation of a Constant Q Spectral Transform. *Journal of the Acoustical Society of America*, 1991.

[7] D. Griffin and Jae Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.

[8] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012.

[9] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[10] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, February 1999.

[11] John C. Duchi, Elad Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, 2011.

[12] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid Search, Random Search, Genetic Algorithm : A Big Comparison for NAS, December 2019.

[13] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband : A Novel Bandit-Based Approach to Hyperparameter Optimization. *J. Mach. Learn. Res.*, March 2016.

[14] Wenbo Zhu, Mou Wang, Xiao-Lei Zhang, and Susanto Rahardja. A comparison of handcrafted, parameterized, and learnable features for speech separation. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, December 2021.

[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 2011.

[16] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, December 1997.

[18] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*. ACL, 2014.

[19] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary Evolution Recurrent Neural Networks, May 2016.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning, 2016.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012.

[22] G.Fleury. *Analyse spectrale. Méthodes non-paramétriques et paramétriques*. Ellipses, June 2023.

[23] Sergey Ioffe and Christian Szegedy. Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, JMLR Workshop and Conference Proceedings. JMLR.org, 2015.

[24] Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. Layer Normalization. July 2016.

[25] Dmitry Ulyanov, A. Vedaldi, and V. Lempitsky. Instance Normalization : The Missing Ingredient for Fast Stylization. *ArXiv*, July 2016.

[26] Michèle Audin. *Analyse complexe*. CNRS, 2011.

[27] Ken Kreutz-Delgado. The Complex Gradient Operator and the CR-Calculus, June 2009.

[28] Andy Sarroff. Complex Neural Networks for Audio. *Dartmouth College Ph.D Dissertations*, May 2018.

[29] Yx Hu, Yun Liu, Shubo Lv, Mengtao Xing, Shimin Zhang, Yihui Fu, Jian Wu, Bihong Zhang, and Lei Xie. DCCRN : Deep Complex Convolution Recurrent Network for Phase-Aware Speech Enhancement. August 2020.

[30] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018.

[31] DeLiang Wang and Jitong Chen. Supervised Speech Separation Based on Deep Learning : An Overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, October 2018.

[32] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Deep learning for monaural speech separation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014.

[33] Francesc Lluís, Jordi Pons, and Xavier Serra. End-to-end music source separation : is it possible in the waveform domain ?, June 2019.

[34] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel Wavegan : A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[35] Daniel Stoller, Sebastian Ewert, and Simon Dixon. *Wave-U-Net : A Multi-Scale Neural Network for End-to-End Audio Source Separation*. June 2018.

[36] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. SEGAN : Speech Enhancement Generative Adversarial Network. August 2017.

[37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, June 2014.

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[39] Joan Serrà, Santiago Pascual, Jordi Pons, R. Oguz Araz, and Davide Scaini. Universal Speech Enhancement with Score-based Diffusion, June 2022.

[40] Xavier Serra and Julius Smith. Spectral Modeling Synthesis : A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition. *Computer Music Journal*, 1990.

[41] Kilian Schulze-Forster, Gaël Richard, Liam Kelley, Clement S. J. Doire, and Roland Badeau. Unsupervised Music Source Separation Using Differentiable Parametric Source Models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[42] Ron J. Weiss and Daniel P. W. Ellis. Estimating Single-Channel Source Separation Masks : Relevance Vector Machine Classifiers vs. Pitch-Based Masking. 2006.

[43] Hakan Erdogan, John R. Hershey, Shinji Watanabe, and Jonathan Le Roux. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015.

[44] Kuldip Paliwal, Kamil Wójcicki, and Benjamin Shannon. The importance of phase in speech enhancement. *Speech Communication*, April 2011.

[45] Anthony P. Stark, Kamil K. Wojcicki, James G. Lyons, Kuldip K. Paliwal, and Kuldip K. Paliwal. Noise driven short-time phase spectrum compensation procedure for speech enhancement. In *Interspeech 2008*. ISCA, September 2008.

[46] John R. Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering : Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016.

[47] Zhuo Chen, Yi Luo, and Nima Mesgarani. Deep Attractor Network for single-microphone speaker separation. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP (Conference)*, March 2017.

[48] Yi Luo and Nima Mesgarani. Conv-TasNet : Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, August 2019.

[49] Yi Luo and Nima Mesgarani. TaSNet : Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation. April 2018.

[50] Timo Gerkmann, Martin Krawczyk-Becker, and Jonathan Le Roux. Phase Processing for Single-Channel Speech Enhancement : History and recent advances. *IEEE Signal Processing Magazine*, March 2015.

[51] Zhong-Qiu Wang, Ke Tan, and DeLiang Wang. Deep Learning Based Phase Reconstruction for Speaker Separation : A Trigonometric Perspective. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[52] Dacheng Yin, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. PHASEN : A Phase-and-Harmonics-Aware Speech Enhancement Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, April 2020.

[53] Donald S. Williamson, Yuxuan Wang, and DeLiang Wang. Complex ratio masking for joint enhancement of magnitude and phase. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016.

[54] Yi Luo, Zhuo Chen, and Takuya Yoshioka. Dual-Path RNN : Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[55] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. Sudo RM -RF : Efficient Networks for Universal Audio Source Separation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, September 2020.

[56] Enric Gusó, Jordi Pons, Santiago Pascual, and Joan Serrà. On Loss Functions and Evaluation Metrics for Music Source Separation. In *ICASSP 2022 - 2022 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2022.

[57] Sercan O. Arik, Heewoo Jun, and Gregory Diamos. Fast Spectrogram Inversion using Multi-head Convolutional Neural Networks. *IEEE Signal Processing Letters*, January 2019.

[58] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, July 2006.

[59] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, May 2001.

[60] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2010.

[61] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. LibriMix : An Open-Source Dataset for Generalizable Speech Separation, May 2020.

[62] Harishchandra Dubey, Vishak Gopal, Ross Cutler, Ashkan Aazami, Sergiy Matusevych, Sebastian Braun, Sefik Emre Eskimez, Manthan Thakker, Takuya Yoshioka, Hannes Gamper, and Robert Aichner. Icassp 2022 Deep Noise Suppression Challenge. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2022.

[63] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech : An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[64] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman. Voxceleb : Large-scale speaker verification in the wild, 2019.

[65] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set : An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017.

[66] Hyeong-Seok Choi, Jang-Hyun Kim, Jaesung Huh, Adrian Kim, Jung-Woo Ha, and Kyogu Lee. Phase-Aware Speech Enhancement with Deep Complex U-Net. December 2018.

[67] Chandan K.A. Reddy, Harishchandra Dubey, Kazuhito Koishida, Arun Nair, Vishak Gopal, Ross Cutler, Sebastian Braun, Hannes Gamper, Robert Aichner, and Sriram Srinivasan. INTERSPEECH 2021 Deep Noise Suppression Challenge. In *Interspeech 2021*. ISCA, August 2021.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[69] Diederik P. Kingma and Jimmy Ba. Adam : A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[70] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *ArXiv*, 2015.

[71] Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. *Advances in neural information processing systems*, 2016.

[72] H. Tagare. Notes on Optimization on Stiefel Manifolds. 2011.

[73] Moritz Wolter and Angela Yao. Complex gated recurrent neural networks. *Advances in neural information processing systems*, 2018.

[74] Michael S. Lewicki. Efficient coding of natural sounds. *Nature Neuroscience*, April 2002.

[75] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R. Hershey. Universal Sound Separation. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, October 2019.

[76] David Ditter and Timo Gerkmann. A Multi-Phase Gammatone Filterbank for Speech Separation Via Tasnet. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[77] Mirco Ravanelli and Yoshua Bengio. Speaker Recognition from Raw Waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, December 2018.

[78] Paul-Gauthier Noé, Titouan Parcollet, and Mohamed Morchid. CGCNN : Complex Gabor Convolutional Neural Network on Raw Speech. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[79] Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. Filterbank Design for End-to-end Speech Separation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[80] Navdeep Jaitly and Geoffrey Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011.

[81] Neil Zeghidour, Nicolas Usunier, Iasonas Kokkinos, Thomas Schaiz, Gabriel Synnaeve, and Emmanuel Dupoux. Learning Filterbanks from Raw Speech for Phone Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018.

[82] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR. September 2014.

[83] Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR. September 2015.

[84] Tara Sainath, Ron J. Weiss, Kevin Wilson, Andrew W. Senior, and Oriol Vinyals. Learning the Speech Front-end with Raw Waveform CLDNNs. In *Interspeech*, 2015.

[85] Yedid Hoshen, Ron J. Weiss, and Kevin W. Wilson. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015.

[86] Shrikant Venkataramani, Jonah Casebeer, and Paris Smaragdis. End-To-End Source Separation With Adaptive Front-Ends. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, October 2018.

[87] Liwen Zhang, Ziqiang Shi, Jiqing Han, Anyan Shi, and Ding Ma. FurcaNeXt : End-to-End Monaural Speech Separation with Dynamic Gated Dilated Temporal Convolutional Networks. In Yong Man Ro, Wen-Huang Cheng, Junmo Kim, Wei-Ta Chu, Peng Cui, Jung-Woo Choi, Min-Chun Hu, and Wesley De Neve, editors, *MultiMedia Modeling*, Lecture Notes in Computer Science, Cham, 2020. Springer International Publishing.

[88] Jingjing Chen, Qirong Mao, and Dong Liu. Dual-Path Transformer Network : Direct Context-Aware Modeling for End-to-End Monaural Speech Separation. October 2020.

[89] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. Attention Is All You Need In Speech Separation. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2021.

[90] Florentin Guth, John Zarka, and Stéphane Mallat. Phase collapse in neural networks. In *International Conference on Learning Representations*, 2022.

[91] Brian R Glasberg and Brian C. J Moore. Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, August 1990.

[92] Helena Peic Tukuljac, Benjamin Ricaud, Nicolas Aspert, and Pierre Vandergheynst. SpectroBank : A filter-bank convolutional layer for CNN-based audio applications. May 2023.

[93] E. Bedrosian. A product theorem for Hilbert transforms. *Proceedings of the IEEE*, May 1963.

[94] Y. Meyer and H. Xu. Wavelet analysis and chirps. 1997.

[95] Eric Chassande-Mottin and Patrick Flandrin. Détection de non-stationnarités. 2004.

[96] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Demucs : Deep Extractor for Music Sources with extra unlabeled data remixed, September 2019.

[97] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013.

[98] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications, April 2017.

[99] F. Chollet. Xception : Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[100] Rachid Riad, Olivier Teboul, David Grangier, and Neil Zeghidour. Learning strides in convolutional neural networks. *ICLR*, 2022.

ECOLE DOCTORALE

Titre : Traitement de la phase des signaux audio dans les réseaux de neurones profonds

Mots clés : Traitement du signal, Apprentissage profond, Signal audio

Résumé : La tâche de séparation de sources sonores d'un enregistrement audio requiert un traitement tout particulier. L'avènement des réseaux de neurones profonds a permis d'améliorer cette tâche au prix d'une complexité computationnelle accrue et d'une opacité des algorithmes. Les interférences induites par ces algorithmes, qu'elles soient parasites ou structurées, peuvent perturber la compréhension du signal, en particulier dans le contexte de la restitution de la voix. Ces problèmes se manifestent particulièrement lors de la transmission de discussions en temps réel, exigeant des mesures de performance pour évaluer les modèles de séparation de sources. Les critères incluent la qualité de reconstruction des pistes individuelles, l'intelligibilité des signaux vocaux, la résilience face aux interférences, et d'autres aspects tels que la réduction des coûts computationnels et l'interprétabilité des traitements. Cette thèse vise à rendre ces modèles plus interprétables tout en atténuant leur coût computationnel, en se concentrant particulièrement sur la modélisation de la phase des signaux. La difficulté actuelle réside dans la modélisation adéquate de cette composante, cruciale pour la compréhension du signal audio. Nous explorerons des stratégies telles que l'utilisation de modèles à valeurs complexes, de représentations invariantes à la phase, et de modèles permettant de s'abstraire de la composante de phase. L'objectif final est de parvenir à des avancées significatives dans la modélisation de la phase des signaux au sein des réseaux de neurones profonds, tout en préservant ou réduisant les coûts computationnels et en améliorant l'interprétabilité des décisions des algorithmes existants.

Title : Phase processing of audio signals in deep neural networks

Keywords : Signal processing, Deep learning, Audio signal

Abstract : The task of separating sound sources in an audio recording requires particular attention. The advent of deep neural networks has improved this task at the expense of increased computational complexity and algorithmic opacity. Interferences induced by these algorithms, whether parasitic or structured, can disrupt the understanding of the signal, especially in the context of voice reproduction. These issues become particularly pronounced during real-time discussions, necessitating performance metrics to evaluate source separation models. Criteria include the quality of reconstructing individual tracks, intelligibility of vocal signals, resilience to interferences, and other aspects such as reducing computational costs and improving interpretability of treatments. This thesis aims to enhance the interpretability of these models while mitigating their computational costs, with a specific focus on modeling the phase of signals. The current challenge lies in finding an appropriate model for this crucial component, essential for understanding audio signals. We will explore strategies such as using complex-valued models, phase-invariant representations, and models allowing abstraction from the phase component. The ultimate goal is to achieve significant advancements in modeling signal phase within deep neural networks, while preserving or reducing computational costs and enhancing interpretability of existing algorithmic decisions.

Institut Polytechnique de Paris
91120 Palaiseau, France