Découverte d'unités linguistiques à l'aide de méthodes d'apprentissage non supervisé

Céline Manenti

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Céline MANENTI

Le 25 mars 2019

Découverte d'unités linguistiques à l'aide de méthodes d'apprentissage non supervisé

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Julien PINQUIER et Thomas PELLEGRINI

Jury

Mme Martine ADDA-DECKER, Rapporteure
M. Yannick ESTèVE, Rapporteur
M. Laurent BESACIER, Examinateur
Mme Lori LAMEL, Examinatrice
M. Julien PINQUIER, Directeur de thèse
M. Thomas PELLEGRINI, Co-directeur de thèse

Remerciements

Je tiens à remercier ma famille pour son soutient, ma mère pour ses allers-retours, sa présence et sa cuisine et mon frère pour la logistique en arrière-plan.

Je remercie aussi mes directeurs de thèse pour leur patience durant cette trèèèèès longue rédaction de mémoire, les quelques thésards apparus quand j'étais encore là et particulièrement Sébastien et Benjamin pour leurs discussions scientifiques toujours motivantes :)!

Je tiens aussi à remercier mon ordinateur Legolas pour sa loyauté et à m'excuser pour les autres gpu de l'équipe parfois monopolisés.

Résumé

La découverte d'unités linguistiques élémentaires (phonèmes, mots) uniquement à partir d'enregistrements sonores est un problème non-résolu qui suscite un fort intérêt de la communauté du traitement automatique de la parole, comme en témoignent les nombreuses contributions récentes de l'état de l'art.

Durant cette thèse, nous nous sommes concentrés sur l'utilisation de réseaux de neurones pour répondre au problème. Nous avons approché le problème en utilisant les réseaux de neurones de manière supervisée, faiblement supervisée et multilingue. Nous avons ainsi développé des outils de segmentation automatique en phonèmes et de classification phonétique fondés sur des réseaux de neurones convolutifs. L'outil de segmentation automatique a obtenu 79% de F-mesure sur le corpus de parole conversationnelle en anglais BUCKEYE. Ce résultat est similaire à un annotateur humain d'après l'accord inter-annotateurs fourni par les créateurs du corpus. De plus, il n'a pas besoin de beaucoup de données (environ une dizaine de minutes par locuteur et 5 locuteurs différents) pour être performant. De plus, il est portable à d'autres langues (notamment pour des langues peu dotées telle que le xitsonga). Le système de classification phonétique permet de fixer les différents paramètres et hyperparamètres utiles pour un scénario non supervisé.

Dans le cadre non supervisé, les réseaux de neurones (Auto-Encodeurs) nous ont permis de générer de nouvelles représentations paramétriques, concentrant l'information de la trame d'entrée et ses trames voisines. Nous avons étudié leur utilité pour la compression audio à partir du signal brut, pour laquelle ils se sont montrés efficaces (faible taux de RMS, même avec une compression de 99%). Nous avons également réalisé une pré-étude novatrice sur une utilisation différente des réseaux de neurones, pour générer des vecteurs de paramètres non pas à partir des sorties des couches mais des valeurs des poids des couches. Ces paramètres visent à imiter les coefficients de prédiction linéaire (Linear Predictive Coefficients, LPC).

Dans le contexte de la découverte non supervisée d'unités similaires à des phonèmes (dénommées pseudo-phones dans ce mémoire) et la génération de nouvelles représentations paramétriques phonétiquement discriminantes, nous avons couplé un réseau de neurones avec un outil de regroupement (k-means). L'alternance itérative de ces deux outils a permis la génération de paramètres phonétiquement discriminants pour un même locuteur : de faibles taux d'erreur ABx intra-locuteur de 7,3% pour l'anglais, 8,5% pour le français et 8,4% pour le mandarin ont été obtenus. Ces résultats permettent un gain absolu d'environ 4% par rapport à la baseline (paramètres classiques MFCC) et sont proches des meilleures approches actuelles (1% de plus que le vainqueur du Zero Ressource Speech Challenge 2017). Les résultats inter-locuteurs varient entre 12% et 15% suivant la langue, contre 21% à 25% pour les MFCC.

Glossaire

ACP	Analyse en Composantes Principales
AE	Auto-Encoder
ASR	Automatic Speech Recognition
BnF	Bottleneck Feature
cAE	correspondance Auto-Encoder
CMVN	Cepstral Mean and Variance Normalization
CNN	Convolutional Neural Network
dAE	denoising Auto-Encoder
DNN	Deep Neural Network
DPC	Density Peak Clustering
DPGMM	Dirichlet Process Gaussian Mixture Model
DPMM	Dirichlet Process Mixture Model
DPMoMM	Dirichlet Process Mixture of Mixture Model
DTW	Dynamic Time Warping
EFR	Eigen-Factor Radial
ELU	Exponential Linear Unit
EM	Expectation Maximization
fMLLR	feature space Maximum Likelihood Linear Regression
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
LDA	Linear Discriminant Analysis
LPC	Linear Predictive Coefficients
LSTM	Long Short Term Memory
LVCSR	Large Vocabulary Continuous SpeechRecognition
MAP	Maximum A Posteriori
MFCC	Mel Frequency Cepstral Coefficients
MLAN	Multi-Level Adaptative Network
MLLR	Maximum Likelihood Linear Regression
MLLT	Maximum Likelihood Linear Transform
MLP	Multi-Layer Perceptron
NMI	Normalized Mutual Information
MPRS	Multilingual Phone Recognition System
NC	Normalizes Cuts
PCA	Principal Component Analysis
PER	Phone Error Rate
PLP	Perceptual Linear Prediction
PRS	Phone Recognition System
QBE	Query-by-Example

RAP	Reconnaissance Automatique de la Parole
RCNN	Recurrent Convolutional Neural Network
ReLU	Rectified Linear Units
RMS	Root Mean Square
RNN	Recurrent Neural Network
sAE	sparse Auto-Encoder
SC	Spectral Clustering
S-DTW	Segmental Dynamic Time Warping
SGD	Stochastic Gradient Descent
SGMM	Subspace Gaussian Mixture Model
SMM	Subspace Mixture Model
SNN	Spiking Neural Network
STD	Spoken Term Detection
TA	Taux d'Apprentissage
TTS	Text To Speech
UBM	Universal Background Model
UTD	Unsupervised Term Discovery
VAD	Voice Activity Detection
VTLN	Vocal Track Lenght Normalization
VTLP	Vocal Track Lenght Perturbation
ZCA	Zero-phase Component Analysis
ZRSC	Zero Resource Speech Challenge

Table des matières

Chapitre 1

Introduction

Les outils de traitement automatique de la parole, tels que les outils de reconnaissance automatique de la parole, sont aujourd'hui de plus en plus performants et robustes. Ils peuvent être utilisés par diverses applications, telles que les assistants vocaux (Alexa, Siri, Cortana et autres (HOY 2018)) qui permettent d'être en interaction avec de nombreux objets connectés comme des téléphones ou des enceintes, par exemple. Cependant, entraîner de tels modèles nécessite de grands corpus annotés manuellement. Des milliers d'heures (3400 heures par exemple dans (Haşim SAK et al. 2015 ; J. LI et al. 2018)), voire des centaines de milliers (150k heures, en situation semi-supervisée (SOLTAU, LIAO et Hasim SAK 2016)) peuvent être utilisées. Obtenir de tels corpus est très coûteux, notamment en temps. Par exemple, il a fallu 5 ans pour le *Spoken Dutch Corpus* possédant environ 1000 heures (SCHUURMAN et al. 2004 ; OOSTDIJK et al. 2002) et une durée similaire (1999-2005) pour le corpus BUCKEYE d'une quarantaine d'heures possédant une transcription phonétique et orthographique (KIESLING, DILLEY et RAYMOND 2006). De plus, obtenir des annotations n'est pas possible pour toutes les langues, certaines langues étant uniquement orales et ne possédant pas de transcription textuelle. C'est pourquoi des expériences sont aussi réalisées pour adapter les modèles à des cas où moins d'heures annotées sont disponibles : par exemple 40 heures dans (HELMKE et al. 2015), 70 heures dans (PELLEGRINI 2008). C'est un sujet actuel qui fait l'objet de beaucoup de travaux de recherche récents (BESACIER et al. 2014). Lorsque peu de données annotées sont disponibles pour une langue, nous parlons alors de langue « peu dotée » (en ressources).

Il existe des travaux pour créer des corpus pour des langues peu dotées, tel que BULB (Breaking the Unwritten Language Barrier) (ADDA, ADDA-DECKER et al. 2016 ; ADDA, STÜKER et al. 2016), afin de réaliser une bonne documentation pour 3 langues Bantu (Basaa, Myene, Embosi). Leur but a été de créer des corpus de plus de 100h pour chaque langue, avec transcription phonétique automatique et traduction française mot à mot, et de joindre à ce travail des outils utiles pour les linguistes. Il existe aussi d'autres projets, comme la collection Pangloss de constitution de corpus de langues orales (MICHAUD et al. 2016). À noter que de nombreux autres projets sont hébergés sur la plate-forme COCOON [1].

Parvenir à obtenir des modèles pour ces langues en se basant sur peu voir aucune ressource est un défi scientifique actuel. Nous devinons qu'apprendre les éléments d'une langue sans l'aide de supervision humaine est un but atteignable puisque tous les enfants réalisent effectivement cette tâche, apprenant naturellement leur langue maternelle en l'écoutant parler autour d'eux. Obtenir le même résultat avec un ordinateur est aujourd'hui un point de recherche important.

Preuve de l'engouement actuel pour ce domaine, le *Zero Resource Speech Challenge* (ZRSC) a vu le jour et a rassemblé plusieurs équipes de chercheurs travaillant

1. https://cocoon.huma-num.fr

sur cette problématique. Une adaptation des modèles acoustiques et de langage déjà existants dans un cadre supervisé à un cadre non supervisé est un des axes de recherche qui en ressort.

C'est pourquoi nous nous sommes intéressé aux langues peu et pas dotées, et plus précisément à la découverte des phonèmes (sous-unités lexicales) dans des enregistrements audio non annotés manuellement dans le but d'apprendre des langues de manière non supervisée.

Arriver à découvrir des connaissances sur des langues peu dotées nécessite le recours à des méthodes non supervisées. Ces méthodes se fondent sur les répétitions des unités et sous-unités lexicales (phonèmes, syllabes, mots) dans la parole. Généralement, les segments de parole proches sont rassemblés et étudiés.

Une méthode de référence dans ce domaine est l'utilisation d'alignements dynamiques des segments de parole (Segmental Dynamic Time Warping, S-DTW), comme cela a été réalisé dans le travail de thèse de Havard (HAVARD 2017) : *Découverte non supervisée de lexique à partir d'un corpus multimodal pour la documentation des langues en danger*. Cet outil est généralement un socle sur lequel sont fondés d'autres outils, comme dans la thèse de Kamper (KAMPER 2017) : *Unsupervised neural and Bayesian models for zero-resource speech processing*. En effet, il utilise des paires de segments de parole regroupés par alignement dynamique pour générer de nouveaux paramètres à l'aide de réseaux de neurones non supervisés, appelés *correspondance Auto-Encoder* (cAE).

Obtenir ensuite des « pseudo-mots »peut se faire à l'aide de modèles Bayésiens non supervisés (KAMPER 2017), permettant d'obtenir une segmentation du signal en segments de parole (GOLDWATER, GRIFFITHS et JOHNSON 2009) puis de regrouper les segments similaires.

Un outil qui a actuellement du succès pour réaliser diverses tâches de traitement automatique de la parole est le réseau de neurones profond (SCHMIDHUBER 2015). Les réseaux de neurones ont maintenant près de 70 ans, mais leur démocratisation est plus récente. Celle-ci a été principalement amenée par l'augmentation des capacités de calcul des ordinateurs, notamment grâce aux cartes graphiques, et leur réussite dans divers challenges et tâches, telle que la reconnaissance de l'écriture manuelle où ils obtiennent moins de 1% d'erreur wan2013regularization. Les réseaux de neurones se sont également montrés efficaces en traitement de la parole, dépassant les modèles historiques (modèles de Markov Cachés fondés sur des Modèles de Mélanges de lois Gaussiennes) en reconnaissance des mots (X. HUANG, BAKER et REDDY 2014), comme illustré dans la figure 1.1.

Ainsi, les réseaux de neurones profonds, et le *deep learning* (apprentissage profond) en général, se montrent très efficaces pour résoudre les problèmes dans un cadre supervisé. Néanmoins, leur adaptation à des tâches non supervisées est encore balbutiante.

Ce travail s'inscrit dans le cadre de l'utilisation de réseaux de neurones pour la découverte de sous-unités lexicales de même que l'apprentissage de représentations phonétiquement discriminantes. L'apprentissage de nouvelles représentations paramétrique est l'utilisation d'outils non supervisés permettant d'obtenir pour chaque trame d'un signal un vecteur de paramètres, autre que les représentations paramétriques déjà existantes (MFCC, ...)

Nous avons construit cette thèse autour des réseaux de neurones et du traitement automatique des phonèmes. **Optimiser des réseaux de neurones pour la classification et la segmentation en phonèmes** est la première tâche que nous réalisons. Les modèles en traitement automatique de la parole sont devenus performants dans un cadre supervisé mais des progrès restent à faire en faiblement supervisé. **Obtenir des**

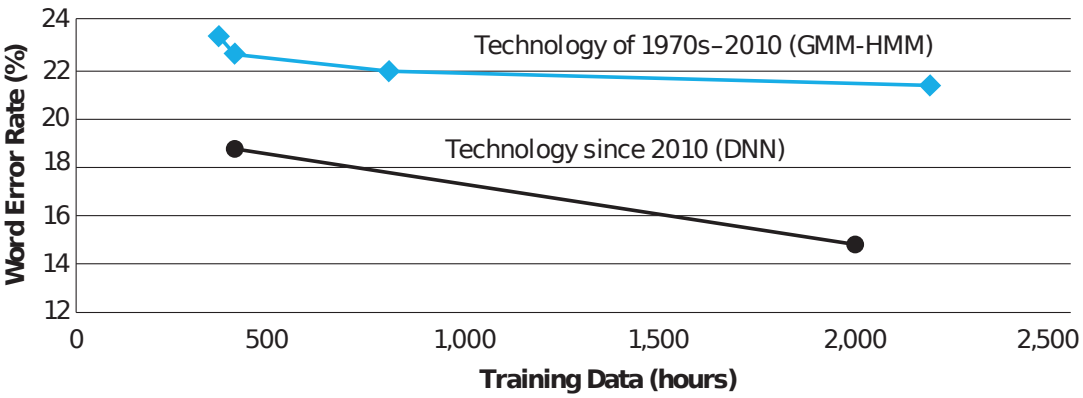

FIGURE 1.1 – Illustration issue de (X. HUANG, BAKER et REDDY 2014) reportant les améliorations apportés par les réseaux de neurones (DNN : Deep Neural Network) dans le domaine de la reconnaissance des mots par rapport à l'approche historique (GMM-HMM : Gaussian Mixture Model - Hidden Markov Model)

modèles robustes à peu de données d'apprentissage ou multilingues est important pour que les modèles puissent être utiles dans le cadre de langues peu dotées.

Les réseaux de neurones sont des outils puissants mais nécessitant généralement beaucoup de données d'apprentissage. **Adapter ces réseaux de neurones à un cadre non supervisé** est une problématique actuelle en recherche, notamment pour la **génération non supervisée de nouvelles représentations paramétriques**. Le but est d'obtenir des représentations plus discriminantes que les paramètres acoustiques communément utilisés dans le traitement de la parole, tels que les bancs de filtres, les MFCC, ... En effet, pour aider les travaux réalisés sur des langues peu voire pas dotées, il est important de pouvoir **découvrir de manière non supervisée les unités et sous-unités de la langue**.

Pour répondre à ces verrous scientifiques et évaluer les approches proposées, nous avons fait attention à la **portabilité** des modèles en les testant sur des langues éloignées : occidentales (anglais, français), africaine (xitsonga), asiatique (mandarin) pour lesquelles nous disposons d'une vérité terrain, et à leur **robustesse** en testant leur comportement avec de faibles quantités de données.

Le document est structuré en 5 chapitres.

Nous commençons cette thèse par un état de l'art sur le traitement automatique de la parole, avec un survol rapide de la reconnaissance automatique de la parole dans un cadre supervisé puis faiblement supervisé avant de s'arrêter plus longuement sur les travaux de recherche effectués dans un cadre non supervisé. Cet état de l'art est suivi des définitions des divers outils et corpora utilisés au cours de cette thèse.

Nous abordons ensuite dans les deux chapitres suivants les expériences supervisées, faiblement supervisées et multilingues. Celles-ci sont réalisées d'une part en classification et d'autre part en segmentation phonétique à l'aide de réseaux de neurones. Les outils ainsi conçus seront testés à l'intérieur de modèles non supervisés dans le dernier chapitre.

Nous abordons dans les deux chapitres suivants la partie non supervisée de cette thèse.

Dans un premier temps, nous expérimentons les réseaux de neurones non supervisés. Nous décrivons trois réseaux de neurones différents. Les deux premiers sont des auto-encodeurs : le premier travaillant à concentrer l'information importante à

partir de paramètres acoustiques et le second réalisant de la compression puis de la synthèse audio directement sur le signal brut. Le troisième réseau part d'une idée différente, imitant des coefficients de prédiction linéaire (Linear Predictive Coefficients, LPC).

Dans un second temps, nous présentons un modèle de découverte de pseudo-phones et de génération non supervisée de nouvelles représentations paramétriques. Nous y testons divers outils et paramètres, notamment ceux décrits dans les chapitres précédents. Nous évaluons notre modèle en terme de découverte non supervisée de sous-unités lexicales et analysons les résultats obtenus sur les données du challenge *Zero Ressource Speech Challenge* (ZRSC).

Nous terminons par une conclusion et proposons quelques pistes pour la poursuite de ce travail.

Chapitre 2

État de l'art, définitions, corpus

Sommaire

Ce chapitre est découpé en trois principales parties : un état de l'art, des définitions, puis une description des corpora. Dans un premier temps, nous commençons par décrire le contexte du traitement automatique de la parole dans un cadre supervisé puis peu supervisé, puis nous nous intéressons à l'état de l'art des approches non-supervisées. Dans un deuxième temps, nous présentons les méthodes de paramétrisation du signal de parole et de modélisation que nous avons utilisées durant cette thèse. Enfin, dans un troisième temps, nous présentons les corpora utilisés durant cette thèse et analysons leurs principales caractéristiques. Tout au long de cet état de l'art, nous nous concentrons sur des unités linguistiques de parole telles que les mots et les phonèmes.

2.1 État de l'art

Nous commençons cet état de l'art par un aperçu très succinct de méthodes supervisées de modélisation d'unités de parole. Nous nous intéressons ensuite à l'objectif de cette thèse : la découverte d'unités sous-lexicales de manière non-supervisée, ainsi que l'apprentissage de représentations ayant un pouvoir discriminant pour cette tâche.

2.1.1 Modélisation supervisée d'unités linguistiques de parole

Plusieurs niveaux peuvent être concernés dans une tâche de modélisation d'unités linguistiques de parole : la phrase, le mot et les unités sous-lexicales (comme les phonèmes).

Reconnaissance automatique de parole continue

L'application la plus connue en traitement de la parole est probablement la Reconnaissance Automatique de la Parole (RAP ou, en anglais : Automatic Speech Recognition, ASR). L'ASR est généralement composé d'un modèle acoustique et d'un modèle de langue. De nombreux outils existent pour réaliser des modèles de reconnaissance automatique de la parole, comme Kaldi (POVEY, GHOSHAL et al. 2011). Kaldi regroupe un ensemble d'outils pour chaque étape, dont la modélisation acoustique. Parmi les outils proposés, nous pouvons citer les mélanges de lois gaussiennes (Gaussian Mixture Models, GMM) (BILMES et al. 1998) et leur version améliorée par l'utilisation de sous-espaces (Subspace Gaussian Mixture Models, SGMM) (POVEY, Lukáš BURGET et al. 2011) ou encore les modèles de Markov cachés (Hidden Markov Models, HMM) (RABINER 1989).

Plus récemment, la renaissance des réseaux de neurones et en particulier des réseaux profonds (Deep Neural Network, DNN) et du *deep learning* en général a débouché sur des progrès significatifs en RAP. Les DNN ont d'abord été utilisés dans des systèmes hybrides HMM-DNN pour calculer les probabilités d'émission des états des HMM (G. HINTON et al. 2012). Plus récemment, des approches dites *end-to-end* essaient de se passer totalement des HMM, comme par exemple les systèmes "DeepSpeech" n'utilisant que des réseaux de neurones (HANNUN et al. 2014) illustrés figure 2.1. L'ajout d'un modèle de langue N-gram améliore néanmoins les résultats de manière significative en forçant à obtenir des phrases plus cohérentes.

La reconnaissance automatique de la parole a de nombreuses applications. Elle peut par exemple servir pour faire du sous-titrage en direct (STADTSCHNITZER et SCHMIDT 2015) ou aussi être utilisée par des outils d'assistance pour conférence (ASSAYAG et al. 2015). Elle peut même servir de support à des études linguistiques, par

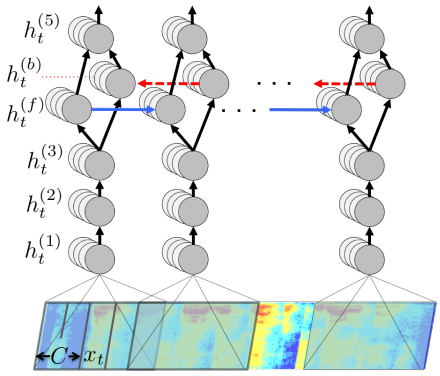

FIGURE 2.1 – Image issue de (HANNUN et al. 2014) illustrant le réseau de neurones profond composant le modèle DeepSpeech effectuant l'ASR, avec un spectrogramme en entrée, $h_t^{(5)}$ la couche de sortie prédisant les caractères et $h_t^{(b)}$ et $h_t^{(f)}$ les couches récurrentes

exemple sur les variations entre les réalisations des phonèmes (VASILESCU, DUTREY et L. LAMEL 2015).

Reconnaissance automatique de mots

La détection de termes parlés (Spoken Term Detection, STD) (MANDAL, K. P. KUMAR et P. MITRA 2014) est une tâche proche de l'ASR : il ne s'agit plus de reconnaître tout ce qui est dit mais seulement de détecter l'utilisation de certains mots. La STD étant une variante de l'ASR, elle peut être réalisée à l'aide de modèles de reconnaissance de la parole, de préférence à large vocabulaire (Large Vocabulary Continuous Speech Recognition, LVCSR) (SAON et CHIEN 2012). Un exemple d'application d'une STD utilisant un système LVCSR est la détection des mots anglais dans des conversations multilingues (MOTLICEK, VALENTE et GARNER 2010).

Les méthodes non supervisées nous intéressent davantage : les techniques Query-by-Example (QBE), qui peuvent utiliser des mesures de similarité basées sur des déformations temporelles (Dynamic Time Warping, DTW) (SRIKANTHAN, A. KUMAR et R. GUPTA 2011 ; SAKOE et CHIBA 1978) au niveau des fenêtres (trames) ou au niveau des segments (WANG, T. LEE et LEUNG 2011), comme illustré dans la figure 2.2. La DTW cherche la distorsion permettant d'obtenir la plus faible distance entre deux segments et permet ainsi de comparer des segments audio de tailles différentes.

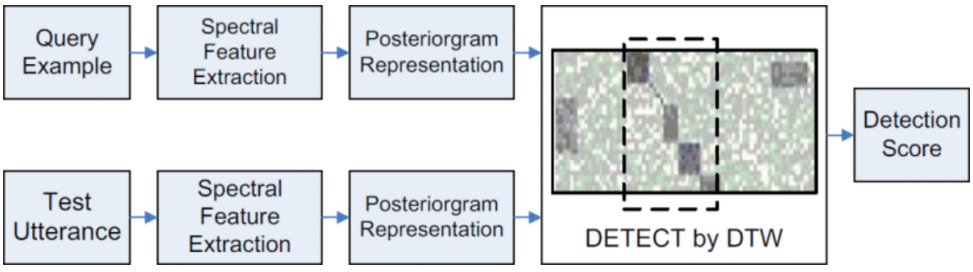

FIGURE 2.2 – Image issue de (WANG, T. LEE et LEUNG 2011) illustrant un système Query-by-Example utilisant une DTW entre deux extraits audio

D'autres techniques utilisant des modèles supervisés peuvent être utilisées, telle que la modélisation de phonèmes (SZOKE, SCHWARZ et al. 2005), notamment basée sur les réseaux de neurones, ou de sous-mots (SZOKE, Lukás BURGET et al. 2008).

Reconnaissance automatique de phonèmes

Les outils de classification des phonèmes peuvent être ensuite utilisés par d'autres tâches, notamment par l'ASR. Selon l'application, il n'est pas nécessaire d'être trop précis sur les différentes prononciations des phonèmes et certaines classes phonétiques proches peuvent être regroupées lors du calcul du taux de classification (ou du taux d'erreur), passant de 61 à 39 classes sur TIMIT et permettant évidemment d'obtenir de meilleurs scores (K.-F. LEE et HON 1989).

Dans le domaine de la classification phonétique, les réseaux de neurones obtiennent parmi les meilleurs scores. Joint à un GMM, un réseau de neurone profond a permis d'obtenir un peu moins de 20% d'erreur phonétique (Phone Error Rate, PER) sur TIMIT (TRAVADI et NARAYANAN 2015). Il existe plusieurs réseaux différents, tels que les réseaux denses (Multi-Layer Perceptron, MLP), les réseaux convolutionnels (Convolutional Neural Network, CNN) (LECUN, BENGIO et al. 1995) et les réseaux récurrents (Recurrent Neural Network, RNN) (DE MULDER, BETHARD et MOENS 2015). Des expériences ont comparé ces différents réseaux durant lesquelles les MLP se sont montrés moins adaptés à la tâche de reconnaissance de phonèmes que les CNN (PALAZ, COLLOBERT et DOSS 2013 ; PALAZ, COLLOBERT et al. 2015). Les RNN permettent quant à eux de prendre un plus grand contexte en considération et permettent d'obtenir seulement 17,7% d'erreur sur un ensemble de test de TIMIT (GRAVES, MOHAMED et G. HINTON 2013). Divers modèles de réseaux récurrents existent, comme ceux utilisant des couches LSTM (ARISOY et SARAÇLAR 2015) ou encore ceux utilisant à la fois des couches de convolution et des couches récurrentes (RCNN) (HU et al. 2015).

Des réseaux moins courants peuvent aussi être utilisés avec succès pour l'ASR, illustrés figure 2.3. Nous pouvons par exemple citer les réseaux d'ondelettes (JEMAI et al. 2015), schéma de gauche. Ce sont des réseaux de neurones de trois couches : une couche d'entrée, une couche cachée dont les neurones sont des fonctions d'ondelettes et une couche de sortie. Nous pouvons aussi mentionner les réseaux à décharge, aussi appelés réseaux impulsionnels ou de spike (Spiking Neural Network, SNN) (LOISELLE 2004 ; LOISELLE et al. 2005 ; TAVANAEI et MAIDA 2017), schéma de droite. Ce sont des réseaux qui « accumulent » les valeurs reçues jusqu'à dépasser un seuil et se « décharger » en envoyant une impulsion.

FIGURE 2.3 – Image de gauche issue de (JEMAI et al. 2015) illustrant l'architecture d'un réseau d'ondelettes, avec (g_i) les versions dilatées et déplacées des fonctions d'ondelettes. Images de droite issue de (TAVANAEI et MAIDA 2017) illustrant la réponse d'un neurone impulsionnel

Quelque soit la méthode utilisée, le choix des paramètres utilisés en entrée est important. Généralement, les probabilités des classes phonétiques prédites par les outils de classification sont calculées à partir de paramètres acoustiques tels que les MFCC (Mel Frequency Cepstral Coefficient) (VERGIN, O'SHAUGHNESSY et FARHAT 1999). Elles peuvent aussi être calculées directement à partir du signal audio brut si l'on utilise un réseau de neurones adapté à cette entrée, comme certains CNN. Ceux-ci parviennent à obtenir près de 70% de classification en phonèmes sur un ensemble de test de TIMIT avec une structure composée de plusieurs couches de convolution suivies de couches denses. Ces résultats sont 2% inférieurs à ceux obtenus avec des MFCC en entrée (PALAZ, COLLOBERT et DOSS 2013). Pour arriver à prédire des classes phonétiques à partir du signal, le CNN peut travailler sur une très petite échelle temporelle de 2 à 4 ms. Les valeurs alors obtenues à la sortie des couches de convolution peuvent ensuite être utilisées comme paramètres généralisables à d'autres corpora (PALAZ, COLLOBERT et al. 2015). Générer de nouveaux paramètres à l'aide de réseaux de neurones peut aussi se faire à partir d'une entrée plus conventionnelle que le signal audio brut. Ainsi, les paramètres issus d'un CNN entraîné sur des bancs de filtres et ajoutés à des i-vecteurs (obtenus à l'aide de GMM) ont permis d'obtenir de bons résultats dans une tâche d'ASR sur de la parole téléphonique bruitée (GANAPATHY et al. 2015), dans le cadre du challenge Aspire (*The IARPA ASPIRE challenge* 2015).

Outre les modèles et paramètres utilisés, le contexte phonétique a aussi une incidence sur les résultats : son utilisation a permis dans des expériences une amélioration des prédictions d'environ 0,5% (Haşim SAK et al. 2015). Une analyse plus précise du degré de contrainte articulatoire (RECASENS, PALLARÈS et FONTDEVILA 1997) montre que les labiales exercent moins d'influence coarticulatrice sur les voyelles voisines que les dentales, permettant une classification des voyelles à côté des labiales plus facile et obtenant des résultats environ 12% meilleurs (DUTTA et PANDEY 2015). Les informations apportées par le contexte phonétique peuvent être utilisés dans les modèles acoustiques, à l'aide d'HMM ou d'arbres de décision (Hainan XU et al. 2015).

Dans le contexte de cette thèse, nous nous intéressons à des situations où les corpus ne sont pas toujours disponibles en quantité et qualité suffisante. L'apprentissage est alors faiblement supervisé.

2.1.2 Modélisation faiblement supervisée

Lorsque l'on travaille sur des langues ayant peu de données annotées, différents cas peuvent se trouver. Nous pouvons utiliser des transcriptions orthographiques et transformer les graphèmes en phonèmes. En effet, l'orthographe (mots) vers l'écriture phonétique permet d'avoir accès à davantage de ressources (M. DAVEL et al. 2015). Pour améliorer les résultats dans le cas d'une langue peu dotée, les deux principales approches sont d'agrandir les données disponibles (approche semi-supervisée) ou d'utiliser d'autres langues possédant des corpus plus conséquents (approche multilingue). Dans le cas où une langue proche très liée est bien documentée, elle peut être utilisée directement pour générer des graphèmes et un dictionnaire avec environ 50% d'erreur au niveau orthographique (STAHLBERG et al. 2014).

À l'échelle du mot

Le *NIST Open Keyword Search Evaluation* (OPENKWS), ayant eu lieu en 2014, 2015 et 2016 (OPENKWS14, OPENSKW15, OPENKWS16)(*NIST Open Keyword Search Evaluation (OpenKWS)* 2013 ; NIST 2013), s'inscrit dans cette optique. Il propose comme corpus d'apprentissage beaucoup de données dans plusieurs langues différentes et peu de données dans la langue cible (quelques heures), en transcription orthographique, avec peu de temps pour entraîner et améliorer le système (3 semaines en 2014, 2 en 2015 puis 1 seule en 2016). Le but de ce challenge s'inscrivant dans le programme Babel IARPA [1] est de créer en peu de temps des systèmes de recherche de mots-clés pour des langues peu dotées.

Le système de base du OPENKWS utilise Kaldi (TRMAL et al. 2017) avec des dictionnaires de prononciation basés sur les graphèmes et non les phonèmes. Il combine trois approches pour la recherche de mots-clés : recherche de séquences de mots similaires (G. CHEN et al. 2013), recherche de séquences phonétiques similaires et recherche de séquences de syllabes ou morphèmes similaires.

Pour répondre à ce problème, des paramètres de Bottleneck multilingues (BottleNeck features, BnF) ont été utilisés à la place des paramètres acoustiques usuels (GOLIK et al. 2015) et ont permis d'obtenir des résultats 10% meilleurs que ceux obtenus avec des paramètres PLP (Perceptual Linear Prediction) (CAI et al. 2015). L'augmentation des données de la langue cible peu dotée, en ajoutant du bruit aux données déjà existantes ou en modifiant les caractéristiques de la parole en perturbant artificiellement la longueur de l'appareil vocal (Vocal Tract Length Perturbation, VTLP (JAITLY et G. E. HINTON 2013)), a également permis d'améliorer les résultats, d'environ 1% (CAI et al. 2015).

À l'échelle du phonème

Un système de reconnaissance des phonèmes idéal serait capable de reconnaître les sons de toutes les langues parlées existantes, comme le peuvent les bébés durant leur première année avant de se spécialiser dans leur langue maternelle (HALLÉ 2004). Faire un système capable de reconnaître les sons de toutes les langues du monde est le but qui motive des projets tels que le *GlobalPhone project* (SCHULTZ, WESTPHAL et A. WAIBEL 1997 ; SCHULTZ et SCHLIPPE 2014). Le *GlobalPhone project* se base sur un corpus multilingue regroupant une grande variabilité de langues parmi les langues les plus courantes. En 1997, le corpus contenait 9 des 12 langues les plus parlées (SCHULTZ, WESTPHAL et A. WAIBEL 1997), puis s'est agrandi jusqu'à en posséder 20 en 2014 (SCHULTZ et SCHLIPPE 2014).

Ce projet a permis le développement de différents systèmes de reconnaissance de phonèmes, indépendants de la langue ou multilingues : des systèmes de reconnaissance des phonèmes (Phone Recognition System, PRS) indépendants de la langue (aussi appelés universels) (SINISCALCHI, LYU et al. 2012) ou multilingues (Multilingual PRS, MPRS) (MANJUNATH, RAO et JAYAGOPI 2017). Essayer de créer un classifieur universel peut se faire en utilisant un réseau de neurone par trait articulatoire à détecter (SINISCALCHI, SVENDSEN et C.-H. LEE 2008). Les probabilités obtenues en sortie pour les langues testés n'ayant pas servie à l'apprentissage sont pertinentes. Un système MPRS peut être composé d'un premier bloc déterminant le langage suivi d'un deuxième bloc réalisant la reconnaissance phonétique (MÜLLER, STÜKER et A. WAIBEL 2018), ou de la même manière une adaptation au locuteur avec un bloc par

1. https://www.iarpa.gov/index.php/research-programs/babel

locuteur (HAMPSHIRE et A. H. WAIBEL 1990). Ce type d'architecture pose trois principaux problèmes : une erreur de reconnaissance de la langue implique une transcription phonétique fausse, le développement de ce système est coûteux (un bloc de reconnaissance phonétique par langue) et exclut les langues peu dotées en ressources. Dans l'idée contraire, un système MPRS peut être un gros bloc commun avec des arbres de décision pour l'attribution des phonèmes à la fin (SCHULTZ et A. WAIBEL 2001). Ce système permet d'obtenir plus facilement de bons résultats lors de l'ajout d'une langue avec peu de ressources grâce au bloc commun.

D'autres méthodes existent pour adapter les systèmes ASR à des problèmes multilingues, comme par exemple en utilisant des modèles de mélanges de lois gaussiennes à sous-espaces (Subspace Gaussian Mixture Models, SGMM) (POVEY, Lukśš BURGET et al. 2010). Ce n'est pas un modèle de phonèmes universels comme le projet *GlobalPhone* mais un modèle de mélange de lois gaussiennes universel dont les langues utilisent un sous-espace (Lukáš BURGET et al. 2010). Dans les cas où peu de données sont disponibles pour la langue cible, les SGMM permettent alors d'obtenir de meilleurs résultats que les GMM, ou que les DNN (pour moins d'1 heure (SAMSON et al. 2015) ou moins de 4 heures (SRIRANJANI, UMESH et al. 2015) de données). Il y a différentes façons d'estimer le sous-espace de la langue cible. Il peut être estimé avec un transfert des sous-espaces partagés entre des langues sources (L. LU, GHOSHAL et RENALS 2011), avec une adaptation de ces sous-ensembles avec le Maximum A Posteriori (MAP) (L. LU, GHOSHAL et RENALS 2012) ou encore avec une combinaison linéaire des sous-espaces des langues sources (Subspace Mixture Model, SMM) (MIAO, METZE et A. WAIBEL 2013).

Malgré tout, apprendre sur un gros corpus et transférer le système obtenu sur un petit corpus peut être une tâche difficile. Dans un cadre supervisé, le taux d'erreur phonétique (PER) avoisine les 20-30%, selon le corpus et le système utilisé. Dans un cadre avec très peu de données, (SCHARENBORG, CIANNELLA et al. 2017) obtient 71% de PER, même si ce score peut être amélioré en utilisant les attributions phonétiques obtenues pour ré-entraîner le réseau. Ces résultats peuvent être davantage améliorés avec un nouvel enregistrement de quelques phrases par des natifs de la langue et leur annotation manuelle : 10% à 15% d'erreur de moins pour 30 minutes d'annotations manuelles (BARTELS et al. 2016).

L'utilisation de paramètres de Bottleneck multilingues semblent être l'une des meilleures pistes actuelles. Ces paramètres sont issus d'une taille inférieur aux couches précédente et suivante. Cette couche peut se situer parmi les dernières couches cachées d'un réseau de neurones multilingue ou multi-domaines (STOLCKE et al. 2006). Ils permettent l'apprentissage rapide d'une nouvelle langue et obtiennent de meilleurs résultats que les paramètres usuels, principalement pour les langues cibles ayant peu de données disponibles mais aussi pour les langues sources ayant servi pour l'apprentissage (J. CUI et al. 2015). Les BnF obtenus peuvent être couplés à d'autres paramètres (ex : MFCC) à l'entrée du deuxième réseau. Ces réseaux sont appelés des réseaux adaptatifs multi-niveaux (Multi-Level Adaptive Networks, MLAN) (LAZARIDIS et al. 2016), comme illustré dans la figure 2.4. À gauche est illustré un réseau MLAN, avec les BnF du premier NN utilisés en entrée du deuxième NN. Le schéma de droite montre un MLAN dont le premier réseau est multitâches, utilisant les classes de la langue cible en plus de celles de la langue source.

Pour améliorer les résultats, un système semi-supervisé (VESELY, HANNEMANN et Lukas BURGET 2013) peut être ajouté au système multilingue (Haihua XU, CHONG et al. 2015 ; SAMSON et al. 2015). Néanmoins, l'amélioration s'avère être très faible lorsqu'il n'y pas pas suffisamment de données (Haihua XU, SU et al. 2016).

La tâche devient encore plus difficile lorsque aucune donnée n'est disponible.

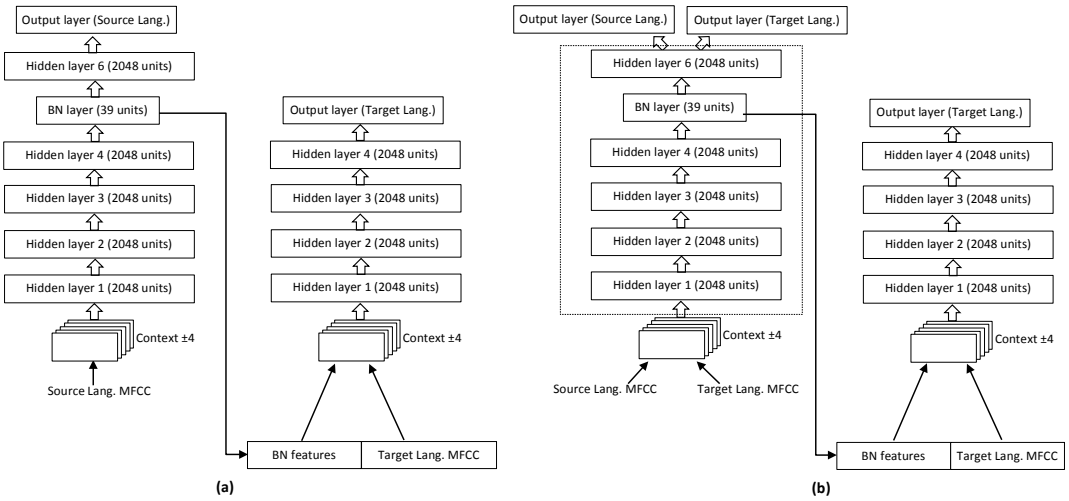

FIGURE 2.4 – Image issue de (LAZARIDIS et al. 2016) illustrant un réseau MLAN (*à gauche*) et un réseau MLAN multitâches (*à droite*)

2.1.3 Modélisation non supervisée

Lorsqu'il n'y a aucune annotation manuelle disponible, les modèles doivent découvrir eux-mêmes les différentes unités et sous-unités composant le langage.

La découverte d'unités linguistiques peut être utilisé par certaines méthodes pour découvrir les sous-unités et à générer de meilleurs paramètres. De nombreuses approches et modèles sont rapportés dans la littérature concernant la découverte d'unités de parole, tels que les dotplots (K. W. CHURCH et HELFMAN 1993), une méthode graphique de comparaison de séquences, et le Segmental-DTW (S-DTW), utilisant la similarité cosinus, qui donne les distances entre les modèles acoustiques phonétiques (JANSEN, K. CHURCH et HERMANSKY 2010 ; PARK et J. R. GLASS 2008), ou encore les cartes auto-organisatrices (MIYAZAWA et al. 2011).

Découverte de pseudo-phones

Découvrir de manière non supervisée les différents phonèmes composant une langue peut s'effectuer à l'aide de diverses méthodes de regroupement, qui travaillent généralement avec des représentations paramétriques de taille fixe.

Pour trouver des segments de tailles variables, l'étape de regroupement est précédée d'une tâche de segmentation phonétique non supervisée (QIAO, SHIMOMURA et MINEMATSU 2008 ; SCHARENBORG, WAN et ERNESTUS 2010 ; WANG, T. LEE, LEUNG, MA et al. 2014). Celle-ci peut être utilisée lors de la découverte des groupes phonétiques (C.-y. LEE et J. GLASS 2012 ; C.-T. CHUNG, CHAN et L.-s. LEE 2013 ; C.-H. LEE, SOONG et JUANG 1988 ; BHATI, NAYAK et MURTY 2017), ou encore être employée par des outils d'annotation de corpus (MUMTAZ et al. 2014). Obtenir une bonne segmentation est important pour ces méthodes, l'utilisation de la segmentation manuelle pouvant améliorer de plus de 10% les résultats (ONDEL, Lukáš BURGET et ČERNOCK 2016).

Mais utiliser des segments de tailles variables en entrée est rarement possible pour les méthodes de regroupements. Des solutions existent pour remédier à ce problème : par exemple l'utilisation de la moyenne des paramètres sur les segments (C.-H. LEE, SOONG et JUANG 1988 ; WANG, LEUNG et al. 2012 ; WANG, T. LEE, LEUNG, MA et al. 2013 ; WANG, T. LEE, LEUNG, MA et al. 2014). Certaines méthodes peuvent aussi s'adapter à des segments de tailles variables. Ainsi, alors que les réseaux de

neurones ont besoin de recevoir des données de taille fixe en entrée, l'utilisation d'un RNN, illustré figure 2.5, permet de générer des vecteurs de paramètres de taille fixe sur des segments de tailles variables (Y.-A. CHUNG et al. 2016).

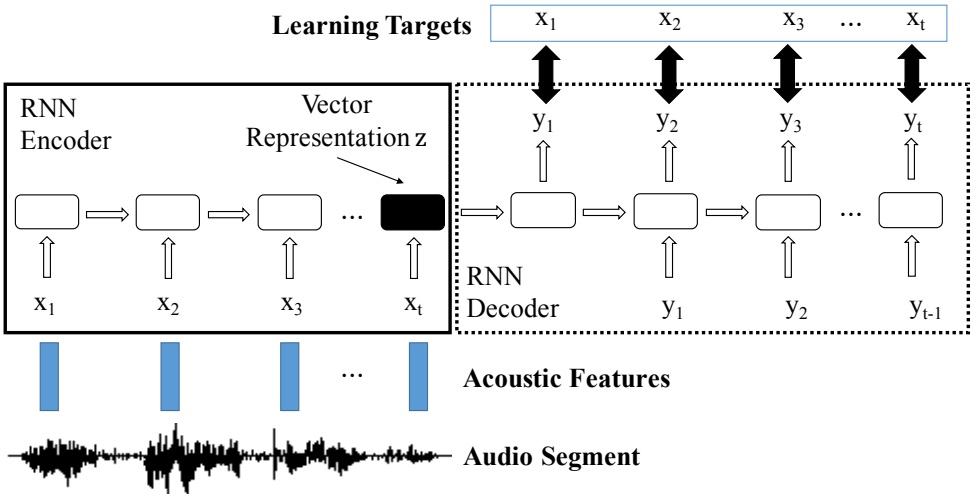

FIGURE 2.5 – Image issue de (Y.-A. CHUNG et al. 2016) illustrant un RNN Auto-Encodeur permettant de générer des vecteurs de paramètres de taille fixe à partir de segments audio de tailles variables

A partir de paramètres de taille fixe obtenus, différentes méthodes de regroupement peuvent être utilisées, que nous présentons dans les prochains paragraphes : réseaux de neurones, k-means, méthodes gaussiennes, bayésiennes, hiérarchiques, ou encore des méthodes moins courantes, telles que les coupes normalisées, les pics de densité ou les codages parcimonieux. Les groupes ainsi trouvés ne correspondent pas exactement aux classes phonétiques et sont généralement appelés pseudo-phones ou sous-unités linguistiques (ou lexicales).

Ces sous-unités peuvent être trouvées en utilisant des méthodes de codage parcimonieux, parfois utilisées pour la modélisation supervisée de la parole (SHARMA et al. 2018). Dans un cadre non supervisé, des méthodes méta-heuristiques peuvent permettre de trouver des groupes obtenant une matrice de confusion visuellement intéressante, avec une diagonale marquée (AGENBAG et NIESLER 2015). Diverses méthodes peuvent être utilisées, telles que des Auto-Encodeur (AE) (G. E. HINTON et SALAKHUTDINOV 2006). Un AE est un réseau de neurones non supervisé apprenant à reconstruire l'entrée après plusieurs projections effectuées par ses couches cachées. Selon le choix de ses différents hyperparamètres, un AE peut générer des vecteurs de paramètres proches de vecteurs binaires, pouvant ainsi être utilisés pour définir des groupes. Cette caractéristique peut être accentuée par des contraintes ajoutées lors de l'apprentissage, certains de ces réseau sont alors appelés des AE parcimonieux (Sparse AE) (TANIGUCHI et al. 2016). Les paramètres générés, une fois binarisés, permettent de regrouper ensemble les segments de valeurs égales. S'il y a trop de groupes ainsi obtenus, une autre méthode de regroupement peut être utilisée pour regrouper les vecteurs les plus proches, telle que les k-means (COATES et NG 2012 ; BADINO, CANEVARI et al. 2014).

Un autre type de réseau peut permettre d'obtenir des regroupements : un réseau de Kohonen (V. MITRA, VERGYRI et FRANCO 2016), prenant des BnF en entrée comme illustré figure 2.6, pour un résultat de 58% de reconnaissance phonétique sur un corpus d'Amharic issu du programme BABEL. Les réseaux de Kohonen sont des

cartes auto-organisatrices : ce sont des réseaux de neurones utilisant la distance euclidienne et non pas le produit scalaire et définissant des liens de voisinage entre les neurones.

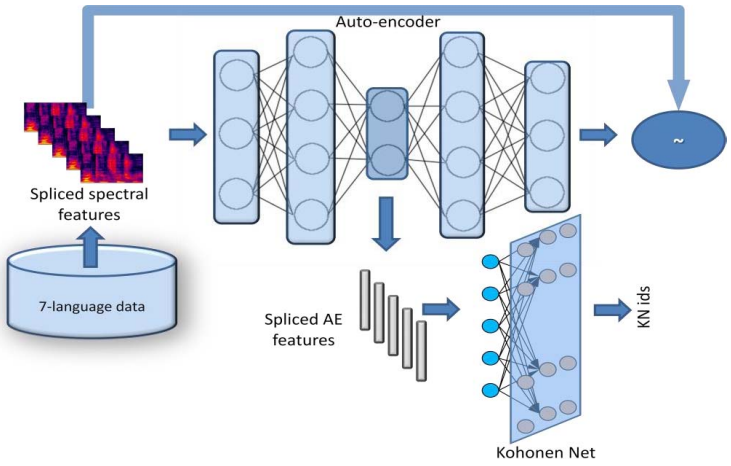

FIGURE 2.6 – Image issue de (V. MITRA, VERGYRI et FRANCO 2016) illustrant les différentes étapes de leur modèle, avec BnF extraits d'un AE utilisés en entrée d'un réseau de Kohonen

De nombreuses approches bayésiennes non paramétriques ont été utilisées pour le regroupement en sous-unités linguistiques. Des DPMM (Processus de Dirichlet avec HMM et GMM, aussi appelés DPGMM) utilisés sur des segments phonétiques découpés de manière non supervisée ont été testés dans (C.-y. LEE et J. GLASS 2012). Leurs résultats, affichés sous forme d'une matrice de confusion, sont visuellement intéressants, comme nous pouvons le voir dans la figure 2.7. Ils montrent une bonne correspondance entre les regroupements effectués et les classes phonétiques, avec souvent plusieurs groupes pour un phone et parfois plusieurs phonèmes dans un même groupe. Une amélioration des DPMM sont les mélanges de mélanges de gaussiennes (Dirichlet Process Mixture of Mixtures Model, DPMoMM). Les groupes représentant les sous-unités lexicales sont eux-mêmes des groupes de groupes, comme pour l'approche supervisée. Ces regroupements sont plus difficiles à déterminer en non supervisé mais différentes méthodes existent (CHANG et FISHER III 2013), permettant de meilleurs résultats, c'est-à-dire principalement moins de groupes différents pour représenter une même classe phonétique (HECK, SAKTI et NAKAMURA 2018a). D'autres améliorations sont possibles. Ainsi, l'approche bayésienne variationnelle permet un entraînement plus rapide que l'échantillonneur de Gibbs grâce à la parallélisation possible de l'algorithme et, en terme d'information mutuelle (S. GUPTA, RAMESH et BLASCH 2008), obtient de meilleurs résultats (ONDEL, Lukáš BURGET et ČERNOCK 2016). L'initialisation à l'aide d'un processus de Dirichlet (ANTONIAK 1974 ; BLEI, JORDAN et al. 2006) peut être améliorée en considérant le contexte phonétique avec un modèle de langage bigramme (ONDEL, Lukaš BURGET et al. 2017).

Les méthodes hiérarchiques ont moins de succès à cause de leur coût élevé qui les rend généralement inutilisables pour de grandes bases de données. Pourtant, des expériences ont montré qu'elles pouvaient obtenir de bons résultats : environ 30% au sens de l'information mutuelle normalisée (Normalized Mutual Information, NMI) sur TIMIT (LERATO et NIESLER 2012). Un regroupement hiérarchique agglomératif multi-stages découpant les données en sous-ensembles a permis de suffisamment améliorer les performances pour rendre l'algorithme applicable à de

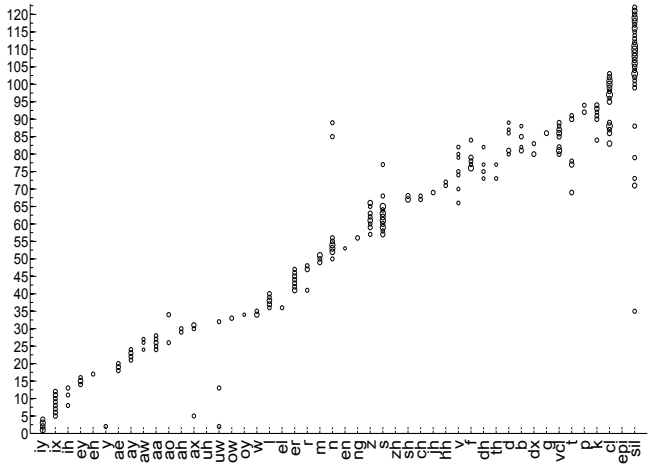

FIGURE 2.7 – Image issue de (C.-y. LEE et J. GLASS 2012) illustrant la pureté des groupes obtenus sous forme de matrice de confusion, avec seulement les paires groupes/phonèmes présentes plus de 200 fois pour plus de clarté

grandes bases de données. La méthode de calcul de similarité inter-groupe utilisée est la méthode de Ward minimisant la variance intra-groupe (MURTAGH et LEGENDRE 2014). Cette méthode a obtenu des résultats similaires aux regroupements spectraux parallèles (W.-Y. CHEN et al. 2011) pour la tâche de regroupement de segments acoustiques (LERATO et NIESLER 2015).

Le regroupement des paramètres peut aussi se faire à l'aide d'un algorithme de coupes normalisées (normalized cuts, NC) associées à un GMM (WANG, T. LEE, LEUNG, MA et al. 2013). Les coupes normalisées sont composées d'une réduction de dimension (à l'aide d'un algorithme basé sur la décomposition en vecteurs propres du Laplacien (BELKIN et NIYOGI 2003)) suivie d'un regroupement à l'aide de k-means. Ce travail a obtenu des résultats supérieurs à la quantification vectorielle et aux GMM sur un ensemble de 6 langues différentes : 30,6% de NMI contre 28,5% (avec la quantification vectorielle) et 27,2% (avec GMM). Basés sur des méthodes similaires mais avec du regroupement spectral (Spectral Clustering, SC) (DHILLON, GUAN et KULIS 2004) au lieu de coupes normalisées et l'utilisation d'une modélisation itérative, 37,4% de NMI a été obtenu sur les mêmes données deux ans plus tard par les mêmes auteurs (WANG, T. LEE, LEUNG, MA et al. 2015). Les NC et le SC utilisent tous deux des méthodes de réduction de dimension issus de la même famille que l'analyse en composantes principales (TORRE FRADE 2008). La plus grande amélioration est l'utilisation d'un modèle itératif qui a permis d'améliorer les résultats de 4%.

Néanmoins, l'indication préalable du nombre de groupes à trouver par le k-means reste un point faible.

L'algorithme de regroupements de pics de densité (Density Peak Clustering, DPC) (RODRIGUEZ et LAIO 2014) n'a pas besoin d'avoir cette indication : il repère les centres des groupes, qui sont caractérisés par des zones de plus grande densité. Pour ce faire, l'algorithme utilise la densité locale de chaque point et sa distance avec les points de plus haute densité. Utilisé à la place du k-means dans l'algorithme des coupes normalisées, il a permis d'obtenir un meilleur taux de NMI sur TIMIT : 36,1% contre 34,8% (J. YU et al. 2015).

Parmi les résultats obtenus dans les différents articles précédents, nous avons

sélectionné ceux utilisant le NMI comme mesure d'évaluation pour afficher des résultats comparables et les avons reporté dans la table 2.1. Trois corpus différents sont utilisés : TIMIT, ZRSC15 et OGI-MLTS. TIMIT est le corpus le plus propre, avec de la parole lue, les deux corpus du Challenge ZRSC15 (Zero Resource Speech Challenge de 2015) sont composés de parole conversationnelle et OGI Multi-Language Telephone Speech Corpus (OGI-MLTS) (MUTHUSAMY, COLE et OSHIKA 1992) contient de la parole téléphonique. Notons que les 41% indiqués pour les DPC sont obtenus à l'aide d'un modèle itératif alternant optimisation des paramètres et DPC. Les DPC seuls obtiennent 37%, ce qui est encore le meilleur score obtenu sur TIMIT.

TABLE 2.1 – Comparaison de méthodes de découverte de sous-unités linguistiques par le taux d'information mutuelle normalisée (NMI)

Méthode de regroupement	Année	Corpus	NMI (%)
Arbres hiérarchiques [2]	2012	TIMIT	30
Coupes normalisées [3]	2013	OGI-MLTS	31
Regroupement spectral [4]	2015	OGI-MLTS	37
DPC [5]	2015	TIMIT	41
DP HMM + modèle bigramme [6]	2017	TIMIT	36
DPMoMM [7]	2018	ZRSC15	29

Pour trouver plus facilement des sous-unités lexicales, utiliser des paramètres discriminants pour cette tâche est essentiel. Ainsi, dans (WANG, T. LEE et LEUNG 2011), les DTW et HMM sont utilisés sur des posteriorgrammes pour trouver des pseudo-mots (C.-y. LEE, O'DONNELL et J. GLASS 2015). Les posteriorgrammes sont les distributions des probabilité des classes, par exemple des phonèmes, en fonction du temps. Ceux-ci peuvent être générés par diverses méthodes de regroupements et de classification, notamment par des réseaux de neurones.

Apprentissage de représentations par réseaux de neurones

Des paramètres efficaces pour représenter les différents phonèmes sont des paramètres éliminant les caractéristiques propres au locuteur (genre, âge, accent, ...), aux conditions d'enregistrements, ... pour ne conserver que l'information discriminante pour les différents phonèmes. Pour trouver de manière non supervisée ces représentations, la structure des réseaux utilisés doit être adaptée à l'apprentissage non supervisé. Ces réseaux apprennent à partir des données qu'ils ont à leur disposition, c'est-à-dire les données d'elles-mêmes et de leur voisinage. Si une méthode d'appariement a préalablement été utilisée, telle que la DTW ou d'autres méthodes de découverte non supervisée de termes (Unsupervised Term Discovery, UTD) (JANSEN et VAN DURME 2011), les réseaux disposent aussi de couples de données. Parmi les différents modèles de réseaux de neurones non supervisés, les principaux utilisés dans le domaine de la découverte automatique de sous-unités de la parole sont les Auto-Encodeurs et les réseaux siamois.

2. (LERATO et NIESLER 2012)
3. (WANG, T. LEE, LEUNG, MA et al. 2013)
4. (WANG, T. LEE, LEUNG, MA et al. 2015)
5. (J. YU et al. 2015)
6. (ONDEL, Lukaš BURGET et al. 2017)
7. (HECK, SAKTI et NAKAMURA 2018a)

De nombreuses expériences ont été réalisées sur les AE pour obtenir de meilleurs paramètres, avec différentes structures dont quelques unes sont montrées dans la figue 2.8, issus de l'article (RENSHAW et al. 2015). Ainsi, l'Auto-Encodeur débruitant (denoising AE, dAE) bruite les données utilisées en entrée pour apprendre à en reconstruire la version non bruitée. Pour transformer les MFCC en paramètres plus discriminants, le correspondance Auto-Encodeur (cAE) utilise des couples de mots alignés pour prendre un segment de parole en entrée et reconstruire le segment de parole correspondant du deuxième mot. Une comparaison de l'efficacité de différents type d'AE (AE, dAE et cAE), effectuée sur les corpus du challenge Zero Resource Speech Challenge 2015 à l'aide du taux d'erreur ABx, a montré que le cAE était le plus adapté à la génération de paramètres discriminants pour la parole. Le cAE a obtenu 21,1% d'erreur ABx sur BUCKEYE, contre 25,3% pour le dAE et 28,6% pour le AE simple.

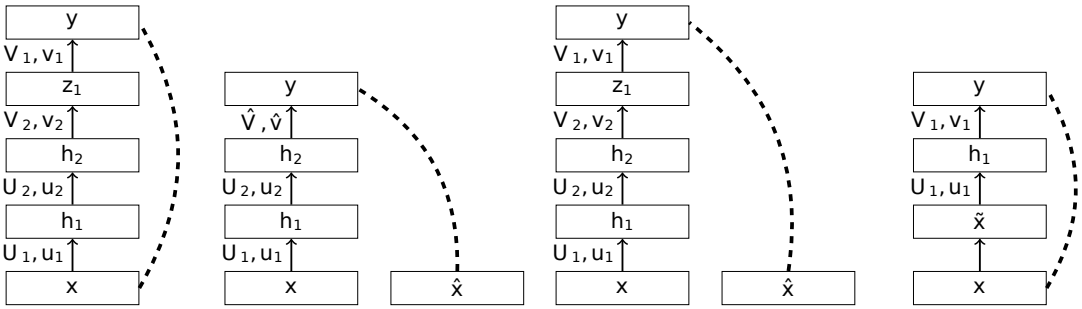

FIGURE 2.8 – Image issue de (RENSHAW et al. 2015) illustrant différentes structures d'AE (de gauche à droite : l'AE profond, le cAE (KAMPER, ELSNER et al. 2015) peu profond, le cAE profond et le dAE (VINCENT et al. 2008))

Un autre réseau de neurones utilisé pour la génération non supervisée de nouveaux paramètres acoustiques sont les réseaux siamois. Ceux-ci se basent sur une idée similaire de celle du correspondance AE en utilisant des couples d'unités linguistiques. La différence est qu'il ne s'agit plus de reconstruire l'autre unité mais que les paramètres générés par les deux entrées appariées génèrent des paramètres proches, comme illustré dans la figue 2.9. Les réseaux siamois peuvent apprendre des représentations à partir de paires de segments phonétiques en rapprochant les paramètres de couples de phonèmes issus d'alignements de couples de mots.

Le réseau ABnet (THIOLLIERE et al. 2015 ; SYNNAEVE et DUPOUX 2015 ; SYNNAEVE, SCHATZ et DUPOUX 2014) est un exemple de réseau siamois construit pour générer des paramètres efficaces pour discriminer les phonèmes. Différentes fonctions de coûts ont été comparées dans (SYNNAEVE, SCHATZ et DUPOUX 2014) pour optimiser au mieux les paramètres générés : la distance euclidienne normalisée, la similarité cosinus carré et un similarité asymétrique cosinus cosinus carré. La distance euclidienne, bien que son principe semble répondre au problème, a obtenus les moins bons résultats : 2% de taux d'erreur ABx supplémentaires aux résultats obtenus avec la similarité cosinus carré et 6% supplémentaire à la similarité cosinus asymétrique (cf. équation 2.1) qui a été réutilisée avec succès dans (THIOLLIERE et al. 2015 ; SYNNAEVE et DUPOUX 2016).

$$coscos^2 = \begin{cases} 1 - cos & \text{si même classe phonétique} \\ cos^2 x & \text{sinon} \end{cases} \tag{2.1}$$

Une récente amélioration a été apportée durant l'apprentissage du réseau en

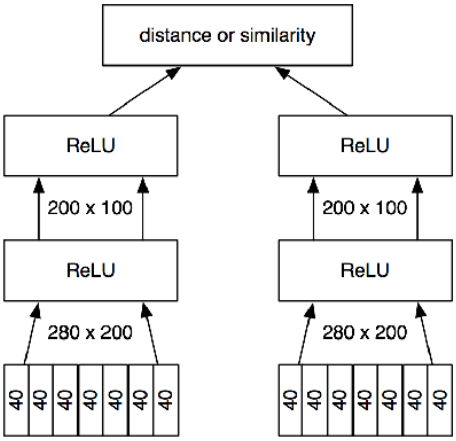

FIGURE 2.9 – Image issue de (SYNNAEVE, SCHATZ et DUPOUX 2014) illustrant la structure d'un réseau travaillant avec des couples de données, appelé réseau siamois

donnant la même importance à tous les mots, qu'ils soient fréquents ou rares, permettant d'améliorer de manière significative les résultats (RIAD et al. 2018). Parallèlement, l'impact de la taille des données d'apprentissage à été étudié dans ce même article. En résultat, les paramètres générés par le réseau siamois se sont révélés plus efficaces que les bancs de filtres, même avec très peu de données.

Les réseaux ne se basent pas tous sur des couples d'unités : certains utilisent des triplets avec deux unités de la même classe et une d'une classe différente. Le coût utilisé permet de rapprocher les paramètres des classes proches et d'éloigner ceux des classes différentes (JANSEN, PLAKAL et al. 2017a ; JANSEN, PLAKAL et al. 2017b). Comme il n'est pas facile de connaître les éléments appartenant au même phonème de manière non supervisée, (SYNNAEVE et DUPOUX 2016) ont considéré les fenêtres proches comme appartenant à la même classe et les fenêtres éloignées (15 à 30 fenêtres) appartenant à des classes différentes en s'appuyant sur les tailles moyennes des phonèmes d'anglais : majoritairement entre 60 et 130 ms (UMEDA 1975 ; UMEDA 1977). Le choix des paramètres d'entrée a de l'influence sur les résultats et l'utilisation de paramètres nommés deep scattering spectrum, utilisant des ondelettes et des filtres et calculés par la ScatNet toolbox (ANDÉN et MALLAT 2014) comme entrée du réseau ABnet à la place des bancs de filtres a permis d'améliorer les paramètres générés en terme de taux d'erreur ABx (-2% sur TIMIT) alors que les deux paramètres ont les mêmes taux d'erreur ABx avant modification par le réseau siamois (ZEGHIDOUR et al. 2016). Ceci fait penser que ces paramètres issus de la ScatNet toolbox sont plus utiles comme entrée des réseaux siamois que les bancs de filtres. Les résultats sur les corpus du ZRSC ont été proches du supervisé pour l'anglais et proches des résultats du gagnant du challenge (H. CHEN et al. 2015).

Mélangeant les deux idées précédentes (le correspondance AE et le réseau siamois), le contrastive AE (ZHENG et al. 2014) est un AE siamois minimisant deux coûts différents : la reconstruction de l'entrée à partir de ces paramètres comme un AE mais en utilisant la ressemblance des paramètres issus d'entrées appariées de la couche centrale dans le calcul du coût (partie siamois), comme illustré dans la figure 2.10. Ce réseau a obtenu 1% de taux d'erreur phonétique de moins que les bancs de filtres.

Plus complexe, un AE récurent (cf. figure 2.11) a permis de générer des paramètres acoustiques pertinents et d'obtenir de bons résultats pour la parole (Y.-A.

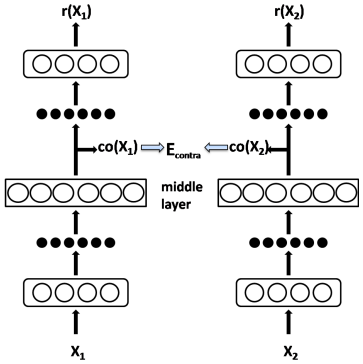

FIGURE 2.10 – Image issue de (ZHENG et al. 2014) illustrant la structure d'un contrastive AE mélangeant réseau siamois et correspondance AE, avec $r(X)$ la partie du coût calculée à partir de l'erreur de reconstruction et $co(X)$ à partir des distances entre les activations issues des 2 entrées différentes

CHUNG et al. 2016) de même que dans plusieurs challenges audio autre que la parole (classification de scènes audio, de sons environnementaux et de genre musicaux) (AMIRIPARIAN et al. 2017) grâce à la boîte à outils auDeep (FREITAG et al. 2017). Ce réseau est inspiré d'un AE LSTM traducteur apprenant à traduire à partir de couples de phrases dans deux langues différentes, prenant en entrée une langue et apprenant à ressortir la phrase traduite (SUTSKEVER, VINYALS et LE 2014).

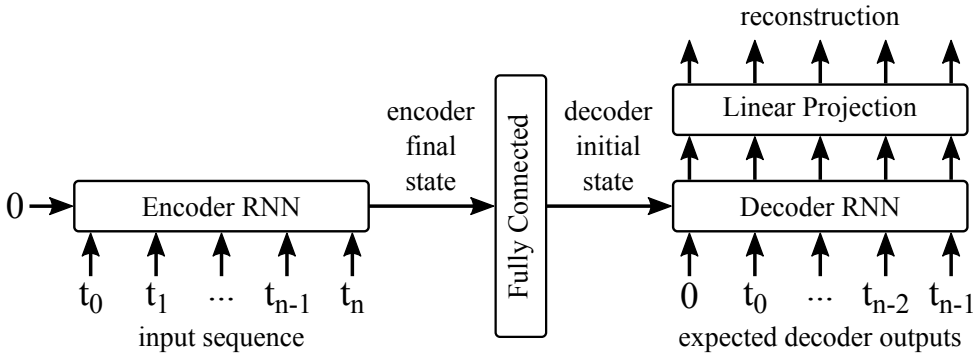

FIGURE 2.11 – Image issue de (AMIRIPARIAN et al. 2017) illustrant la structure d'un réseau AE récurent

Certains de ces réseaux de neurones ont été testés dans des conditions similaires dans le challenge ZRSC 2015. Comme nous l'avons dit plus haut, c'est le cAE qui a obtenu les meilleurs résultats parmi les AE avec 21% d'erreur, mais ces résultats sont inférieurs à ceux obtenus par les différentes versions des réseaux ABnet. Les résultats sont comparés dans la table 2.2. Tous les modèles présentés dans cette table obtiennent de meilleurs résultats que les paramètres de base (MFCC) et des résultats égaux ou légèrement moins bons (selon le corpus testé) que les meilleurs paramètres proposés lors du challenge : les DPGMM.

Le challenge *ZRSC* de 2017, dont la première tâche est la découverte de nouvelles représentations phonétiquement discriminantes, a permis à des systèmes bien plus complexes de voir le jour.

TABLE 2.2 – Comparaison de différents taux d'erreur ABx inter-locuteurs obtenus par des réseaux de neurones non supervisés sur les corpora utilisés lors du challenge ZRSC 2015

Réseau	Anglais	Xitsonga
Baseline (MFCC) [8]	28,1	33,8
DPGMM [9]	16,3	17,2
cAE [10]	21,1	19,3
ABnet [11]	17,9	16,6
ABnet avec paramètres ScatNet [12]	17,0	15,8

Apprentissage de représentations : contexte du Zero Resource Speech Challenge

La première tâche du challenge *ZRSC* est de générer de manière non supervisée de nouveaux paramètres phonétiquement discriminants. Leur aptitude à éloigner les phonèmes de classes différentes et à rapprocher ceux de même classe est évaluée par le taux d'erreur ABx (SCHATZ, PEDDINTI, BACH et al. 2013 ; SCHATZ, PEDDINTI, CAO et al. 2014). Ce taux d'erreur compare les échantillons trois par trois, avec deux éléments d'une même classe et le troisième élément d'une autre classe phonétique. Ce taux d'erreur, davantage détaillé dans le chapitre 6, est découpé en deux mesures dans le challenge : les taux d'erreur intra-locuteur et inter-locuteurs. Ces deux mesures permettent de voir la robustesse des paramètres aux variations dues au locuteur.

Lors des deux éditions du challenge, diverses méthodes ont été utilisées. En 2015, les meilleurs résultats ont été obtenus en utilisant des méthodes de mélanges de gaussiennes utilisant des processus de Dirichlet (Dirichlet Process Gaussian Mixture Models, DPGMM) (H. CHEN et al. 2015). Les DPGMM, aussi appelés modèles de mélanges de gaussiennes infini, ont l'avantage d'estimer eux-mêmes le nombre de regroupements effectués. L'article compare l'implémentation des DPGMM avec des GMM de différents nombres de groupes et montre que les DPGMM obtiennent 1,5% d'erreur de moins pour l'intra-locuteur et quelques dixièmes de pourcentages en moins pour l'inter-locuteurs que le meilleur GMM. Les DPGMM ont été réutilisés avec succès en 2017 par les deux meilleurs participants (HECK, SAKTI et NAKAMURA 2017 ; H. CHEN et al. 2017).

Comme vu dans la section précédente, deux types de réseaux de neurones ont été testés en 2015 (VERSTEEGH, ANGUERA et al. 2016) : un siamois (basé sur le réseau ABNet présenté précédemment dans la section 2.1.3) utilisant des couples de segments obtenus par DTW (THIOLLIERE et al. 2015) et des auto-encodeurs (RENSHAW et al. 2015 ; BADINO, MERETA et ROSASCO 2015). D'autres expériences ont été réalisées avec un modèle itératif alternant un DNN multitâche et des HMM sans parvenir à dépasser les meilleurs résultats (C.-T. CHUNG, TSAI, C.-H. LIU et al. 2017). D'autres paramètres ont également été testés : les paramètres articulatoires (BALJEKAR et al. 2015). L'Auto-Encodeur (AE), de même que les paramètres articulatoires, n'ont pas répondu au problème avec succès en 2015. En 2017, un système utilisant

8. (VERSTEEGH, ANGUERA et al. 2016)
9. (H. CHEN et al. 2015)
10. (RENSHAW et al. 2015)
11. (THIOLLIERE et al. 2015)
12. (ZEGHIDOUR et al. 2016)

deux réseaux de neurones, dont un AE, a permis d'obtenir des paramètres plus discriminants au sens du taux d'erreur ABx (YUAN et al. 2017). Le premier réseau est un DNN multilingue apprenant les groupes trouvés par une méthode de regroupements non supervisés obtenus par DPGMM monolingues sur chacun des corpus. Les paramètres issus d'une couche de Bottleneck du réseau permettent de trouver des paires de mots, en utilisant notamment la DTW, et les aligner pour obtenir des couples de trames. Un AE est ensuite entraîné en utilisant ces paires de mots et la distance euclidienne comme coût de reconstruction. Les paramètres finaux proposés au challenge sont issus d'une de ses couches cachées qui ont permis une amélioration de quelques dixièmes de pourcentages par rapport aux BnF du premier réseau, qui étaient déjà efficaces.

Dans l'édition de 2017 [13] (DUNBAR et al. 2017), les meilleurs résultats du challenge ont été obtenus en suivant le schéma global suivant : regroupement en pseudo-phones des segments de parole puis utilisation d'une classification supervisée pour générer de nouveaux paramètres. Plus précisément, la table 2.3 décrit les étapes des deux meilleurs systèmes. Ils utilisent des DPGMM pour l'étape de regroupement (clustering). Un modèle utilisant des GMM a obtenu des résultats légèrement moins bons (ANSARI et al. 2017).

TABLE 2.3 – Récapitulatif des deux premiers systèmes du ZRSC2017

	Premier : Heck [14]	Deuxième : Chen [15]
Paramétrisation	PLP CMVN VAD UBM VTLN	MFCC + fbank + f0 CMVN VAD VTLN
Regroupements	DPGMM (super-clusters)	DPGMM
Classification	Modèle monophone (+ alignements) Modèle triphone (+ alignements) Estimation LDA+MLLT+fMLLR n-grammes	DNN multilingue

Les paramètres acoustiques utilisés sont les PLP (pour le premier arrivé) ou les MFCC (pour le deuxième), sur lesquels des méthodes d'adaptation au locuteur ont été appliquées à l'aide de kaldi (POVEY, GHOSHAL et al. 2011) : Analyse Linéaire Discriminante (Linear Discriminant Analysis, LDA), transformation selon un critère de maximum de vraisemblance (Maximum Likelihood Linear Transform, MLLT) et maximisation de la vraisemblance de l'espace des paramètres par régression linéaire (feature space Maximum Likelihood Linear Regression, MLLR (SAON, ZWEIG et PADMANABHAN 2001)) pour la soumission arrivée première au challenge (HECK, SAKTI et NAKAMURA 2016b ; HECK, SAKTI et NAKAMURA 2016a ; HECK, SAKTI et NAKAMURA 2017), comme illustré dans la figure 2.12 ; normalisation de la longueur du conduit vocal (Vocal Tract Length Normalization, VTLN) (WEGMANN et al. 1996)

13. http://sapience.dec.ens.fr/bootphon/2017/page_5.html
14. (HECK, SAKTI et NAKAMURA 2017)
15. (H. CHEN et al. 2017)

et normalisation de la moyenne et de la variance Cepstral (Cepstral Mean and Variance Normalization, CMVN) (HAEB-UMBACH 1999) pour le second. Les classes ainsi obtenues sont ensuite utilisées pour entraîner une méthode de classification supervisé. Le premier a utilisé un modèle acoustique suivit d'un modèle de langue qui lui ont permis de corriger les pseudo-phones précédemment obtenus puis de les utiliser pour appliquer des méthodes d'adaptation au locuteur « supervisées » avant d'appliquer un nouveau DPGMM sur ces paramètres. Le deuxième a quant à lui utilisé un réseau de neurones multilingue dont les premières couches sont communes à toutes les langues et dont la dernière couche cachée génère les paramètres soumis au challenge.

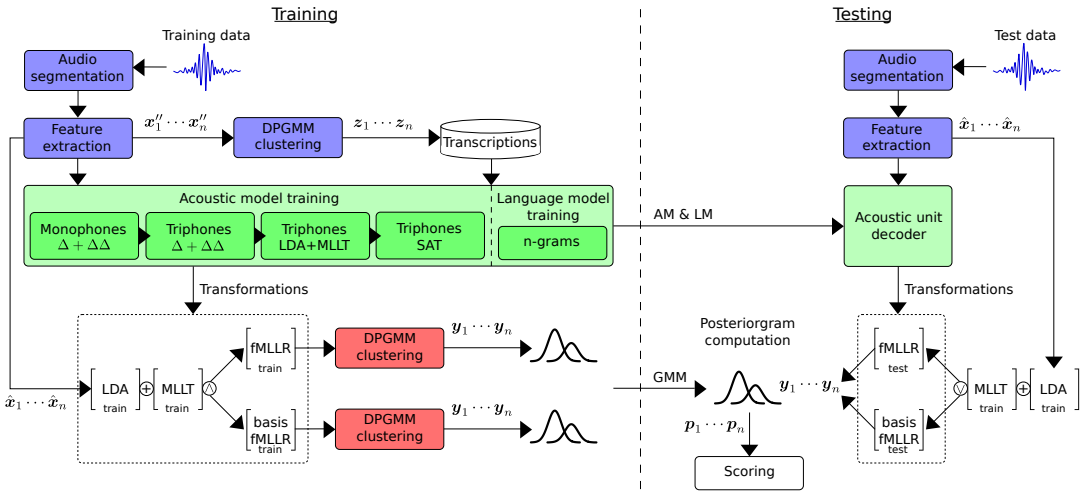

FIGURE 2.12 – Image issue de (HECK, SAKTI et NAKAMURA 2017) illustrant le système de génération non supervisée de paramètres ayant obtenu les meilleurs scores au ZRSC 2017

Les paramètres obtenus en sortie sont une concaténation des prédictions des DPGMM. Ils sont parcimonieux, avec une dimension de l'ordre du millier.

Ces précédents résultats ont été dernièrement légèrement améliorés (S. FENG et T. LEE 2018), notamment en utilisant une supervision issue d'autres langues, riches en annotations, pour entraîner les méthodes d'adaptation au locuteur (VTLN, LDA, MLLT, fMLLR). Le reste du système se base sur une idée similaire aux précédents systèmes étudiés : DPGMM servant à entraîner un classifieur (ici DNN) qui génère de nouveaux paramètres (BnF issus d'une couche cachée). En utilisant la même idée d'adaptation au locuteur, mais apprise de manière supervisée sur d'autres langues, la seule utilisation d'un DPGMM sur les données du *ZRSC* de 2015 a permis de bons résultats (HECK, SAKTI et NAKAMURA 2018b).

2.1.4 Conclusion

Nous avons vu dans cet état de l'art beaucoup d'auteurs utiliser des réseaux de neurones pour travailler sur la parole et c'est également le principal outil employé lors de cette thèse. Dans le cadre supervisé, les réseaux de neurones peuvent réaliser toutes les tâches. Cependant, ils sont plus rarement utilisés seuls dans le domaine non supervisé. Ainsi, ils sont souvent couplés à des méthodes de regroupement, comme nous l'avons nous-même appliqué dans la suite de cette thèse.

Parmi tous les systèmes non supervisés précédemment présentés (générations de paramètres, regroupements en pseudo-phones), certains permettent de mutuellement s'aider et s'améliorer en étant alternés dans un modèle itératif : de meilleurs

groupes permettant d'améliorer les paramètres et de meilleurs paramètres permettant d'améliorer les regroupements. Quelques exemples d'amélioration due aux itérations ont été donnés dans des articles : +4,4% de NMI (J. YU et al. 2015) et +3% de NMI (WANG, T. LEE, LEUNG, MA et al. 2015) pour la découverte de pseudo-phones, et -0,5% de taux d'erreur ABx (HECK, SAKTI et NAKAMURA 2016a) pour la génération de représentation paramétrique discriminante.

Nous allons maintenant présenter les différents outils (paramètres, modèles, ...) utilisés durant cette thèse.

2.2 Définition des paramètres et des modèles

Pour travailler sur la parole, nous avons utilisé différents paramètres (MFCC, bancs de filtres), différents outils de regroupement (k-means, GMM) et de classification (réseaux de neurones), que nous présentons dans cette section.

2.2.1 Paramètres audio

Durant cette thèse, nous avons utilisés principalement deux types de paramètres, adaptés au traitement de la parole : les bancs de filtres et les MFCC. Seules quelques expériences ont été réalisées directement sur le signal brut.

Le signal brut est assez difficile à exploiter directement : les valeurs qui le composent dépendent principalement de la phase des fréquences présentes et du bruit. C'est pourquoi il est utile de réduire la dimension du signal en paramètres discriminants pour la tâche à effectuer.

Les bancs de filtres et les MFCC concentrent l'information du signal en une quarantaine de paramètres (39 bancs de filtres + intensité ou 13 MFCC + dérivées premières et secondes). Cette réduction permet de diminuer la variabilité de même que la redondance du signal et rend le traitement des données plus facile et rapide.

Signal brut

Pour être étudié, le signal de parole est découpé en segments de courtes durées, aussi appelés fenêtres. La taille de ces fenêtres dépend de la tâche effectuée. Ici, nous étudions les phonèmes de la parole.

Pour qu'il n'y ait pas de discontinuité au bord des fenêtres (si la transformée en fréquences considère le segment de signal comme répété indéfiniment), le contenu du segment est multiplié par une fenêtre dont les valeurs sont nulles ou faibles sur les bords, comme par exemple une fenêtre en cosinus. Et pour ne manquer aucune information présente aux bords des fenêtres, un recouvrement est généralement utilisé.

Lorsqu'il y a fenêtrage puis reconstruction du signal, la fenêtre utilisée pour un recouvrement de moitié est la fenêtre en cosinus carré (Hamming, Hanning), car elle permet d'obtenir un signal reconstruit identique à l'original : $sin^2x + cos^2x = 1$. Pour un recouvrement de trois quart ce sont des fenêtres en cosinus 4.

Le signal brut est une suite d'échantillons de son évoluant dans le temps. Les données audio que nous avons utilisées ont un échantillonnage de 16000 valeurs par seconde. Classiquement, nous extrayons les données du signal sur des segments de 25 ms, avec un pas de 10 ms. Les recouvrements les plus fréquents sont de moitié ou de 3/4. Quelques exemples de 4 phonèmes différents ([n], [ɪ], [s] et [ə]) sont illustrés dans la figure 2.13. Nous voyons que la fricative ([s]) est très bruitée par rapport aux

autres phonèmes. Ces exemples sont issus du locuteur *s01* du corpus BUCKEYE (cf section 2.3).

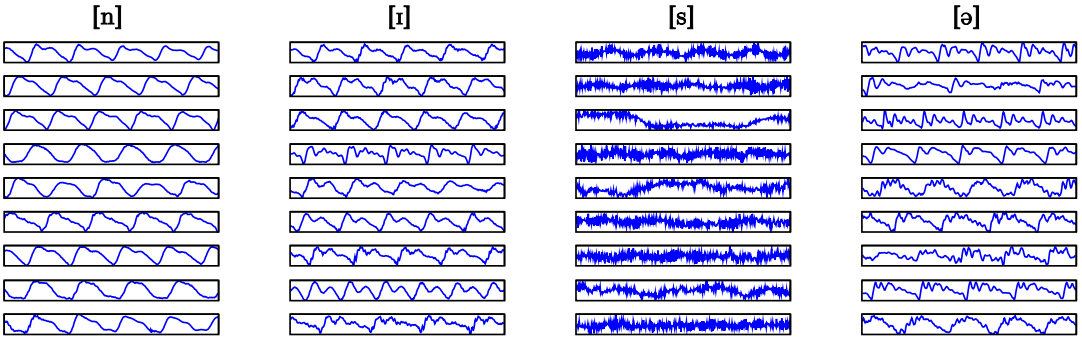

FIGURE 2.13 – Exemples issus du corpus anglais BUCKEYE d'évolution temporelle d'échantillons de différents phonèmes : [n], [ɪ], [s] et [ə] sur une durée de 25 ms

Le signal brut est rarement directement utilisé en entrée d'un outil (de classification, de segmentation ou autre) même si des expériences sont faites dessus (PALAZ, COLLOBERT et al. 2015). Ainsi, dans certains travaux de recherche récents, les réseaux de neurones travaillent directement sur le signal brut. L'avantage de son utilisation est qu'il n'y a pas de perte de données. Mais son utilisation présente de gros inconvénients : une grande sensibilité au bruit, une grande taille de données et une dépendance des valeurs à la phase du segment, notamment. Ceci force les outils à effectuer un travail supplémentaire pour transformer ces valeurs en paramètres adaptés à la tâche à accomplir.

Spectre

La transformée en fréquences la plus connue est la transformée de Fourier : elle projette les signaux dans la base des exponentielles complexes. La transformée de Fourier d'un signal quelconque réel est complexe et symétrique, et peut être décomposée en deux signaux réels : le rayon et l'angle. Nous acceptons une perte d'information en nous intéressons à la valeur absolue du spectre, ne prenant en compte les intensités des fréquences plus que leur phase.

Théorème de Fourier Sous certaines conditions de dérivation et de continuité, tout signal à temps continu s(t) périodique, de période T, peut s'écrire sous la forme d'une somme de signaux sinusoïdaux. Cette somme peut s'écrire de deux manières : forme trigonométrique réelle, forme exponentielle complexe. La version discrète est donnée dans l'équation 2.2.

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{-2i\pi k \frac{n}{N}} \tag{2.2}$$

D'autres décompositions proches existent, comme la décomposition en cosinus (Discret Cosine Transform, DCT) qui est réelle et non symétrique. Des exemples de signaux et de DCT correspondante sont illustrés dans la figure 2.14. Nous voyons que les spectres ont l'avantage d'être parcimonieux pour des signaux faiblement bruités, comme le signal réel illustré par les deux graphiques de droite.

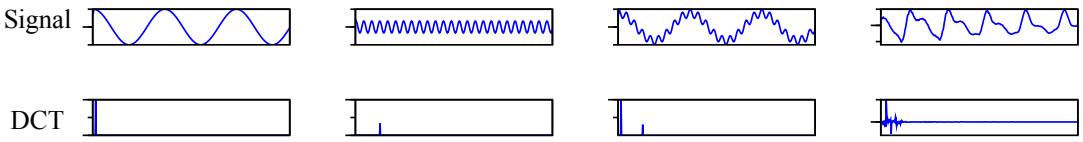

FIGURE 2.14 – Exemples de couples signal/spectre. À gauche, deux signaux contenant une seule fréquence, puis la combinaison de ces deux fréquences. À droite, un signal réel (une réalisation d'un [n])

Bancs de filtres

Les bancs de filtres ont la particularité d'avoir des paramètres corrélés en temps et en fréquence et ainsi d'être particulièrement adaptés en entrée d'un CNN (nous les avons d'ailleurs fortement utilisés dans cette thèse). Le découpage du spectre en bancs de filtres Mel est illustré dans la figure 2.15. La parole et le spectre audible se situent surtout dans les basses fréquences, qui sont privilégiées en étant moyennées sur de plus petites bandes. Les bandes conservant l'information issue des plus hautes fréquences sont les plus larges car les moins importantes du point de vue auditif et donc proportionnnellement les plus bruités. D'autres échelles existent mais l'échelle Mel est la plus couramment utilisée.

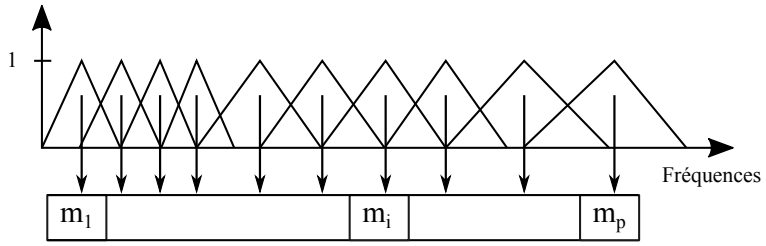

FIGURE 2.15 – Découpage des fréquences en bandes suivant l'échelle Mel pour le calcul des bancs de filtres

Chaque valeur issue d'un banc de filtre correspond à l'intensité du spectre pour le filtre correspondant. Nous utilisons le logarithme de ces valeurs. Les bancs de filtres permettent la construction d'un spectrogramme temps/fréquence, dont plusieurs exemples sont donnés dans la figure 2.16 : chaque colonne correspond à un phonème différent. Nous voyons sur les exemples donnés que le [s] est le plus présent dans les hautes fréquences. Les fréquences des réalisations du [ɪ] et du [ə] se répartissent davantage sur le spectre.

Pour réaliser ces expériences, nous avons tout d'abord utilisé MIRtoolbox (LARTILLOT, TOIVIAINEN et EEROLA 2008) avec MATLAB, puis Speechpy (TORFI 2017; TORFI 2018) en Python. Spectral Python (SPy) (*Welcome to Spectral Python (SPy)* p.d.) nous permet de générer les paramètres de base du Challenge ZRSC17. Pour chacun, nous avons conservé les valeurs par défaut des paramètres.

MFCC

Les MFCC (Mel Frequency Cepstral Coefficients) sont calculés à partir des bancs de filtres : c'est la transformée en cosinus discrète du log-mel-spectre. Pour les obtenir, nous avons utiliser les mêmes outils que pour les bans de filtres, en laissant toujours les paramètres par défaut.

Les MFCC sont généralement plus adaptés aux méthodes utilisant moins l'information apportée par la corrélation des paramètres, tels que les k-means et les GMM

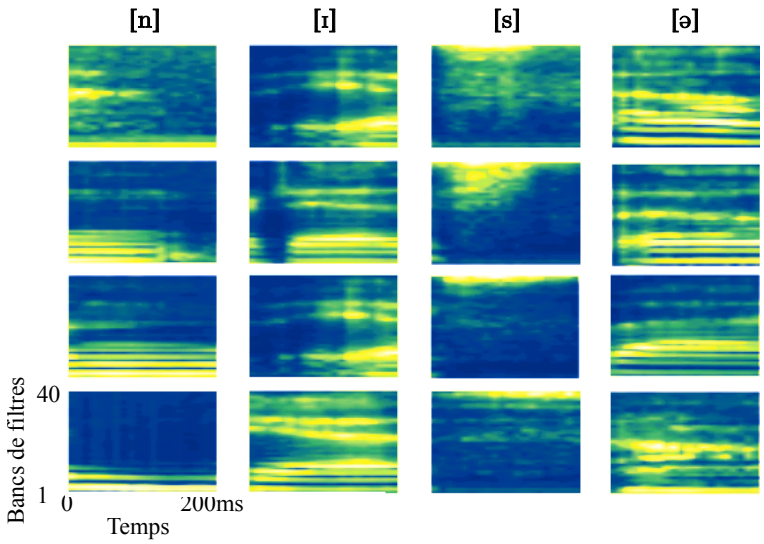

FIGURE 2.16 – Exemples de spectrogrammes de 40 bancs de filtres standards suivant l'échelle mel pour les phonèmes suivants, de gauche à droite : [n], [ɪ], [s] et [ə], sur une durée de 200 ms, issus du corpus anglais BUCKEYE

(à matrice de covariance diagonale) par exemple. C'est grâce à leur pouvoir discriminant qu'ils se montrent efficaces pour des méthodes simples. La figure 2.17 illustre la pertinence de ces paramètres : nous voyons que les exemples projetés (toujours appartenant aux mêmes quatre classes phonétiquement éloignées : le [n], [ɪ], [s] et [ə]) se distinguent sur les graphiques. Pour des raisons de lisibilité, nous avons choisi de n'afficher qu'un nombre limité d'exemples par classe (200). Les différentes classes sont beaucoup moins distinguables sur les plans 2D avec toutes les données.

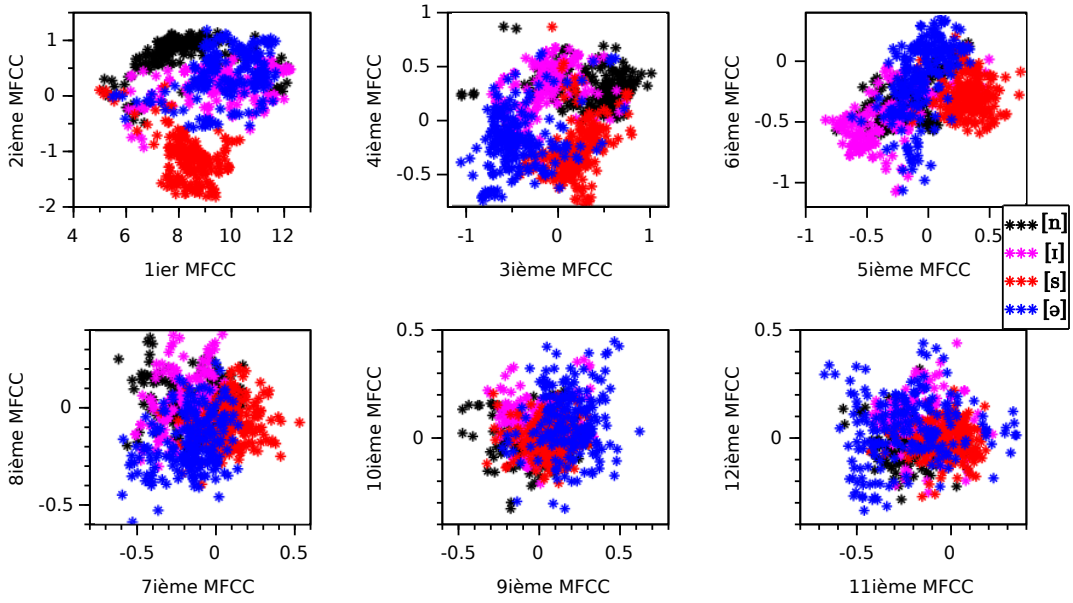

FIGURE 2.17 – Projection de 200 réalisations de chacun des 4 phonèmes [n], [ɪ], [s] et [ə] sur les axes formés par les MFCC

La propriété fondamentale du cepstre est de transformer les convolutions en additions. La représentation cepstrale permet ainsi de dissocier la source du filtre au

conduit. Ceci sépare les valeurs propres aux locuteurs (liées à la fréquence fondamentale et au conduit) à celles propres au message. Les MFCC sont généralement utilisés avec leurs deux premières dérivées temporelles de même que l'énergie du signal.

2.2.2 Normalisations du signal et modification des paramètres

Durant notre thèse, nous avons utilisé des outils non supervisés de normalisation des paramètres. Nous présentons ici l'ACP et la ZCA, les deux principales techniques utilisées durant cette thèse. Les autres méthodes, utilisées ponctuellement, sont décrites lors de leur apparition dans le manuscrit.

Analyse en Composantes Principales

L'Analyse en Composantes Principales (ACP ou, en anglais : Principal Component Analysis, PCA) (ESCOFIER et PAGÈS 2008) est l'algorithme le plus connu. C'est une méthode non supervisée de réduction d'axes qui peut être utilisée aussi bien en classification non supervisée que supervisée. L'ACP cherche les axes principaux des données à analyser pour les projeter dessus (voir algorithme 1). Ces axes forment le sous espace maximisant la variance des données projetées. Plus précisément, considérons les données disposées en matrice M, avec chaque ligne un exemple et chaque colonne un descripteur. Alors les axes principaux sont les vecteurs propres de la matrice de covariance S de M associés aux valeurs propres les plus élevées.

Algorithm 1 Algorithme de l'ACP sur les données X

$S \leftarrow$ matrice de covariance de X
$U \leftarrow$ vecteurs propres de S
$U_{reduit} \leftarrow$ conservation des premiers vecteurs propres
$Z \leftarrow$ projection de X sur l'espace défini par U_{reduit}

La projection des données sur les trois premiers axes de l'ACP sont donnés dans la figure 2.18, avec à gauche les résultats pour les MFCC, au milieu les résultats pour les MFCC centrés et normés et à droite les résultats pour les bancs de filtres. Avec les MFCC bruts, les classes sont plutôt bien séparées, contrairement aux bancs de filtres où les classes sont mélangées. Les classes sont moins distinctes lorsque les MFCC sont normalisés. Nous pouvons avancer comme explication que les premiers MFCC suffisent à faire une première séparation importante des classes et ce sont aussi eux qui ont les valeurs les plus élevées. Normaliser les données réduit leur intensité et donc leur importance dans le calcul de l'ACP.

La pertinence des MFCC, de même que la faible corrélation de leurs paramètres entre eux, peuvent être vues dans le graphique de gauche de la figure 2.19 : les 13 premiers vecteurs correspondent beaucoup à des vecteurs unitaires laissant presque intacts les 13 premiers MFCC. Les autres vecteurs propres mélangent les dérivées premières et secondes. Les vecteurs propres correspondant aux bancs de filtres (graphique de droite) sont totalement différents. Nous voyons apparaître une bande diagonale qui correspond à la décorrélation par l'ACP des fréquences proches.

Zero-phase Components Analysis

Normaliser des paramètres peut se faire à l'aide d'une Zero-phase Components Analysis (ZCA) (KESSY, LEWIN et STRIMMER 2018). La ZCA se base sur la PCA mais

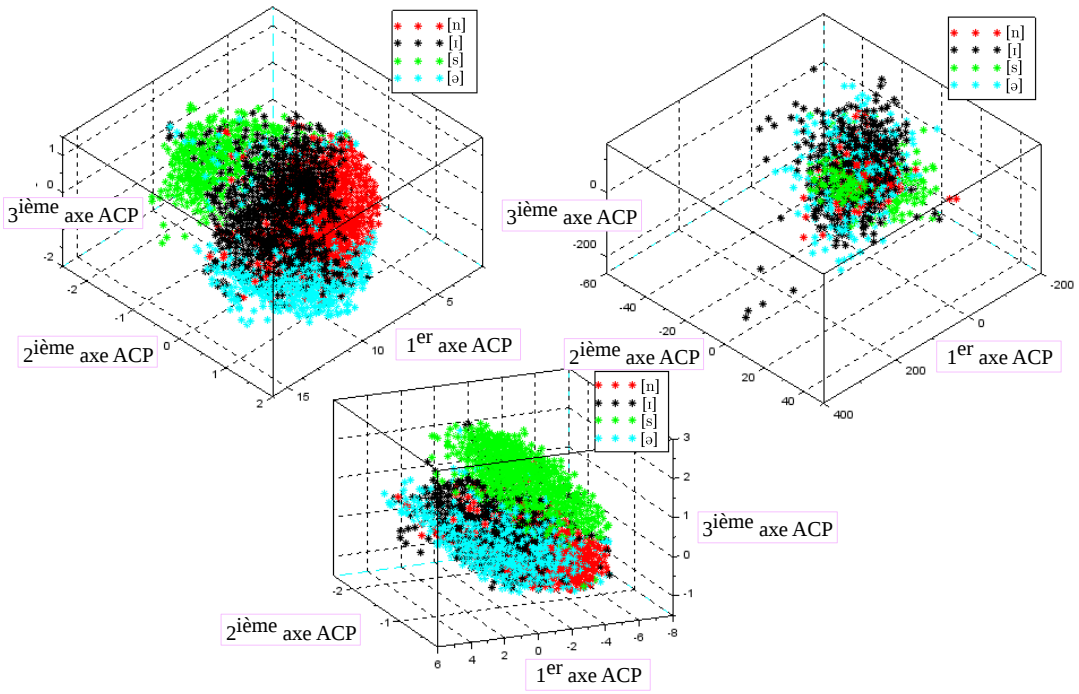

FIGURE 2.18 – Projection des réalisations du locuteur *s01* des 4 phonèmes [n], [ɪ], [s] et [ə] sur les axes formés par les MFCC (*à gauche*), MFCC centrée normés (*au milieu*) et les bancs de filtres (*à droite*)

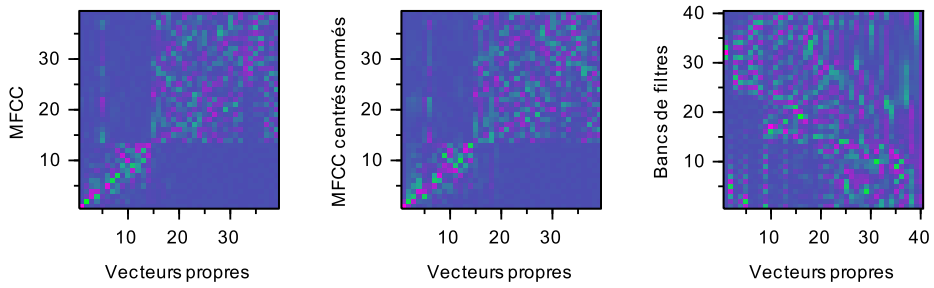

FIGURE 2.19 – Matrice des vecteurs propres correspondant à l'ACP réalisée sur les MFCC (à gauche) et les bancs de filtres (à droite)

conserve la taille et l'orientation des données. La conservation de l'orientation des données permet de calculer et d'appliquer la ZCA localement, par exemple locuteur par locuteur dans notre cas. Cela permet ainsi d'obtenir des paramètres plus robustes au locuteur.

Il a été montré que les k-means pouvaient être plus performants lorsque l'entrée avait été préalablement blanchie par ZCA (COATES et NG 2012) et cette méthode de blanchiment a été utilisée avec succès en traitement de la parole (GWON et al. 2017).

La ZCA (Zero Components Analysis, ZCA) est une technique de blanchiment des paramètres souvent utilisée en traitement des images (COATES et NG 2012). Son but est de normaliser et de blanchir les paramètres tout en conservant leur orientation spatiale d'origine, comme illustré dans la figure 2.20 [16]. La seule différence entre la ZCA et la PCA est la rotation des données.

16. code disponible : https://github.com/topel/demo-zca

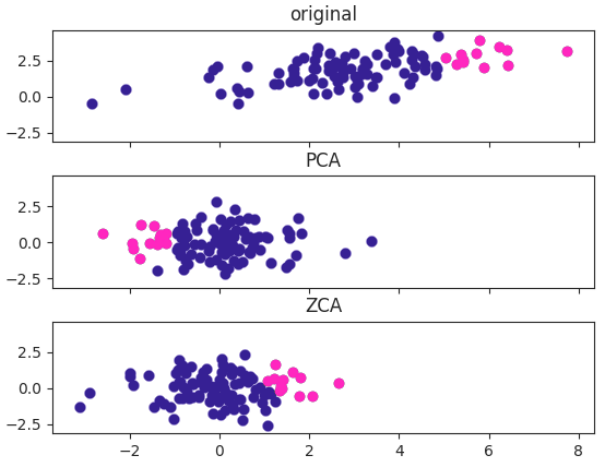

FIGURE 2.20 – Comparaison entre la PCA et la ZCA sur un ensemble de données artificielles 2D montrant la conservation de l'orientation spatiale par la ZCA

Plus précisément, considérons les données disposées dans une matrice X, avec chaque ligne une donnée et chaque colonne un paramètre. $\overline{X}$ correspond à X centré. Soit U, D une décomposition singulière de $\text{cov}(\overline{X})$, la matrice de covariance de $\overline{X}$, avec U la matrice des vecteurs propres et D la matrice diagonale des valeurs propres correspondantes, telles que $\text{cov}(\overline{X}) = UDU^t$. Alors X est blanchi à l'aide de la formule suivante :

$$X_{\text{ZCA}} = U(D+\epsilon)^{-1/2}U^t\overline{X} \tag{2.3}$$

Pour blanchir les données à l'aide de la ZCA, l'algorithme 2 est utilisé en pratique.

Algorithm 2 Algorithme de la ZCA sur les données X

$S \leftarrow$ matrice de covariance de X
$U \leftarrow$ vecteurs propres de S
$U_{seuil} \leftarrow$ seuillage doux de U : pour chaque vecteur propre p, $U_{seuil_p} = U_p/(\sqrt{v_p+\epsilon})$
$U_{oriente} \leftarrow$ rétablissement de l'orientation : $U_{seuil} * U^T$
$Z \leftarrow$ projection de X sur l'espace définit par $U_{oriente}$

La valeur ϵ choisie a un impact important sur les résultats. Nous avons étudié cet impact chapitre 6. Nécessaire pour éviter d'éventuelles divisions par zéro, cet hyperparamètre a aussi pour effet de minimiser l'importance des axes dont la valeur propre est faible.

À notre connaissance, la ZCA n'est pas couramment utilisée en traitement de la parole, et même plus rarement en reconnaissance de la parole. Nous avons trouvé des études de reconnaissance du locuteur utilisant le blanchiment par PCA appliqué aux i-vecteurs, une transformation appelée facteur radial propre (Eigen-Factor Radial, EFR) (BOUSQUET, MATROUF, BONASTRE et al. 2011).

2.2.3 Méthodes de regroupement

Dans le cadre de cette thèse, nous avons besoin de méthodes non supervisées pour découvrir les sous-unités lexicales. Pour cela, nous utilisons des méthodes de regroupement. Nous avons utilisé principalement des k-means mais nous avons aussi testé des GMM. Les sous-sections suivantes décrivent ces deux méthodes.

k-means

Le principe d'un regroupement par k-means (ou k-moyennes en français) est de regrouper itérativement des exemples similaires selon une distance (euclidienne par exemple) à des points (appelés centroïdes) dont les coordonnées sont les coordonnées moyennes des éléments du groupe. Pour cela, deux étapes sont alternées : le calcul des centroïdes (la moyenne des éléments du groupe) et l'attribution des groupes (pour chaque exemple, le groupe dont le centroïde est le plus proche). Le détail est donné dans l'algorithme 3.

Algorithm 3 Algorithme du k-means

$K \leftarrow$ initialisation aléatoire des k centroïdes
Tant que (condition) :
. $C \leftarrow$ attribution des classes les plus proches
. $K \leftarrow$ moyennes des échantillons de chaque classe

Les centroïdes finissent par se diriger vers des zones plus denses (qui ont davantage de poids dans le calcul de la moyenne) tout en s'espaçant les uns des autres, comme nous pouvons le voir dans l'exemple donné figure 2.21. Sur ces données artificielles, les cinq classes sont trouvées au bout de deux itérations.

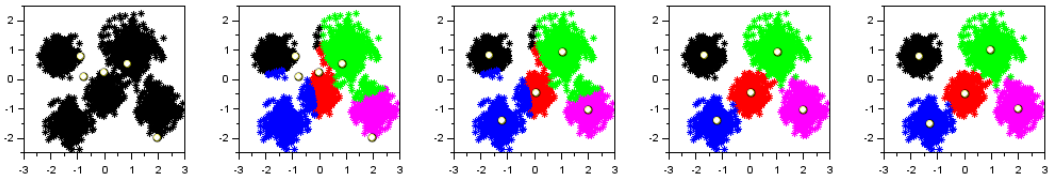

FIGURE 2.21 – Évolution étape par étape des centroïdes (ronds blancs) et attributions (couleurs) des classes du k-means sur un jeu de données artificielles

Les résultats peuvent varier selon l'initialisation des centroïdes choisie. Nous pouvons choisir d'effectuer plusieurs k-means sur les données et sélectionner celui qui est le mieux selon un critère (minimisation le l'inertie par exemple). Il existe des variantes au k-means telles que les k-medians, calculant les centroïdes à l'aide d'une médiane et non d'une moyenne.

Plusieurs outils existent pour effectuer un regroupement par k-means. Comme nous utilisons beaucoup de données, nous avons utilisé FAISS, une bibliothèque performante pouvant travailler sur GPU [17]. FAISS propose aussi le k-means sphérique, utilisant la distance cosinus et non la distance euclidienne.

17. https://github.com/facebookresearch/faiss

GMM

Les GMM sont des modèles de mélange de lois gaussiennes (Gaussian Mixture Model, GMM) (BILMES et al. 1998) considèrent que les échantillons des classes suivent une répartition gaussienne. Dans le cas supervisé, plusieurs gaussiennes peuvent être attribuées à chaque classe pour représenter au mieux sa répartition. Plus exceptionnellement, cela peut aussi être le cas en non supervisé (HECK, SAKTI et NAKAMURA 2018a) mais l'estimation des paramètres est plus complexe.

L'estimation des paramètres (moyennes et variances) se fait généralement à l'aide de l'algorithme d'Estimation-Maximisation (Expectation-Maximization, EM) (BILMES et al. 1998) : il s'agit d'une alternance de l'estimation des classes des échantillons et de l'estimation des paramètres des classes (voir algorithme 4). À la différence du k-means où seules les distances sont utilisées, les classes sont définies par un centre et une variance et les probabilités pour chaque échantillon d'appartenir à une classe est calculée en fonction.

Algorithm 4 Algorithme EM

$K \leftarrow$ initialisation aléatoire des k moyennes
$V \leftarrow$ initialisation des écarts-types des classes par vecteurs unitaires
Tant que (condition) :
. $C \leftarrow$ attribution des classes les plus probables (distance/écart-type)
. $K, V \leftarrow$ moyennes et écarts-types des échantillons de chaque classe

Lorsque les matrices des variances des classes sont fixées diagonales, les groupes ont moins de liberté et ne permettent pas de capter les corrélations entre les différents paramètres. Cependant, utiliser ces matrices diagonales est parfois préférable, par exemple en cas de peu de données pour réduire le nombre de paramètres à estimer. Une illustration des formes des groupes d'un GMM est donnée dans la figure 2.22.

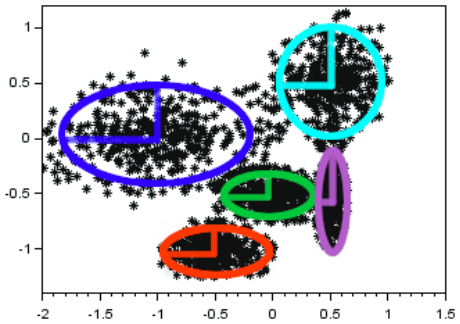

FIGURE 2.22 – Exemples de formes de regroupements d'un GMM à variances diagonales (ovales de couleurs) sur un jeu de données artificielles

2.2.4 Méthode de classification : réseaux de neurones

Nous avons axé cette thèse sur l'utilisation des réseaux de neurones. Ce choix s'est fait dès le début de la thèse au regard de leurs résultats au niveau de l'état de

l'art dans beaucoup de tâches du traitement automatique de la parole et des possibilités diverses qu'ils offrent, avec les différentes sortes de réseaux qui existent. De plus, c'est un domaine très actif qui nécessite encore énormément de recherche.

Introduction

Les réseaux de neurones artificiels sont des successions de projections non linéaires permettant d'atteindre un espace où les valeurs des données répondent au critère demandé durant l'apprentissage du réseau.

Les premiers réseaux de neurones apparaissent vers la fin des années 50 avec le perceptron (ROSENBLATT 1958). À l'époque, les ordinateurs étaient moins puissants et les résultats étaient moins probants que maintenant. Mais les résultats théoriques étaient déjà très encourageants. Ainsi, il a été montré qu'un réseau neuronal à 2 couches avec des fonctions d'activation non linéaires (comme par exemple la *sigmoïde*) peut être considéré comme un approximateur universel de fonction (CYBENKO 1989).

L'apprentissage d'un réseau de neurones s'effectue en faisant remonter les erreurs dans ses couches pour modifier en conséquence les poids attribués aux entrées de chaque couche. L'idée est similaire aux méthode de sélection de paramètres types enveloppeurs (HALL et SMITH 1999), qui permettent de changer l'ensemble de paramètres de départ en fonction des erreurs.

De nombreux réseaux de neurones différents existent, certains souvent utilisés et d'autres plus expérimentaux, comme par exemple les réseaux d'Ondelettes (ZHANG et BENVENISTE 1992), dont les valeurs des poids des neurones sont les valeurs des ondelettes. Ceux-ci ont notamment été utilisés pour la reconnaissance de la parole (JEMAI et al. 2015). Nous pouvons aussi citer l'ACP neuronale (CHITROUB 2004), effectuée par un réseau à une couche, linéaire, et règle d'apprentissage Hebbienne (MILLER et MACKAY 1994), qui approxime la vraie ACP.

Des articles permettent de bien connaître les réseaux de neurones et les multiples modèles et variantes qui existent (SCHMIDHUBER 2015 ; W. LIU et al. 2017). Dans cette sous-section, nous donnerons les principales définitions et outils utilisés durant cette thèse :

- **les neurones**, qui génèrent une sortie à partir des entrées,
- **les couches de neurones**, composées de plusieurs neurones utilisant les mêmes entrées,
- **les différents architectures** (de réseaux de neurones), composées de différentes couches,
- **l'apprentissage** des réseaux,
- **les librairies Python** permettant d'utiliser des réseaux de neurones.

Les neurones

Un neurone effectue une projection dans un espace 1D en effectuant un produit scalaire entre les entrées et ses poids, auquel il applique un biais puis une fonction non linéaire (voir figure 2.23).

Calcul effectué par un neurone :

$$a = f(wp - b) \tag{2.4}$$

avec :

- **Entrées, sortie** : p, a Les entrées sont les paramètres d'entrée du réseau dans le cas de la première couche, les sorties de la couche précédente pour les autres

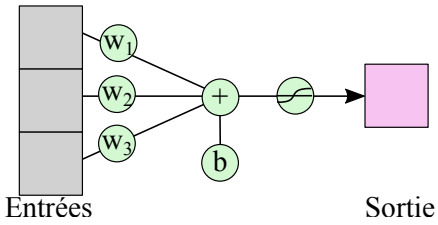

FIGURE 2.23 – Schéma d'un neurone. Une projection non linéaire avec biais (b) : les entrées (carrés gris) sont multipliées par les poids du neurone (w_i) puis sommées

couches. La sortie est obtenue après multiplication des entrées par les poids, ajout du biais puis application de la fonction de transfert (aussi appelée fonction d'activation).

— **Fonction de transfert**, aussi appelé **fonction d'activation** : f La première fonction de transfert utilisée fut un seuil qui transformait la sortie en une sortie binaire, mettant à 1 toutes les valeurs supérieures au seuil. Depuis, d'autres fonctions ont été utilisées, dont des exemples sont illustrés en figure 2.24, permettant un apprentissage plus facile et de meilleurs résultats. Certaines de ces fonctions ont été comparées dans la littérature sur des problèmes spécifiques (DING, H. QIAN et ZHOU 2018) pour lesquels la rectification linéaire a été la plus efficace.

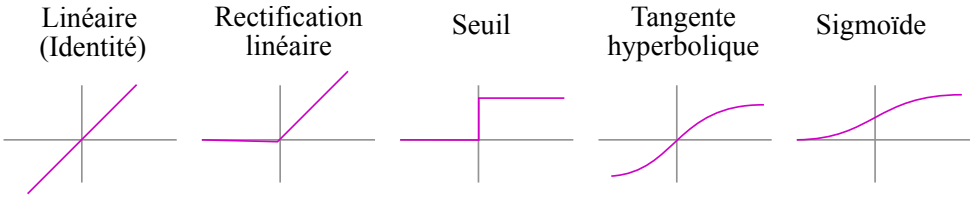

FIGURE 2.24 – Illustration de différentes fonctions de transfert

Les formules des fonctions d'activations présentées sont les suivantes :

- **Identité** ou **linéaire**. Elle est par exemple utilisée par des réseaux non supervisés (CHITROUB 2004) et parfois nécessaire pour la couche de sortie de certains d'entre eux pour pouvoir générer une large gamme de valeurs positives et négatives en sortie, comme par exemple pour certains AE.

$$f(x) = x \tag{2.5}$$

- **Rectification Linéaire** (Rectified Linear Units, ReLU) (DAHL, SAINATH et G. E. HINTON 2013). À la différence d'une fonction linéaire, les valeurs négatives sont mises à zéro.

$$f(x) = \begin{cases} x & \text{si x>0} \\ 0 & \text{sinon} \end{cases} \tag{2.6}$$

- **Marche** ou **seuil**. C'est la fonction d'activation utilisée par les premiers réseaux de neurones. Depuis, des versions dérivables telles que la tangente hyperbolique et la *sigmoïde* sont privilégiées.

$$f(x) = \begin{cases} 1 & \text{si x>0} \\ 0 & \text{sinon} \end{cases} \tag{2.7}$$

- **Tangente hyperbolique**.

$$f(x) = tanh(x) \tag{2.8}$$

- ***sigmoïde***.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

— **Paramètres** appris par le réseau : w les poids (ou pondérations des entrées), b le biais du neurone (aussi appelé seuil d'activation). L'initialisation des poids des neurones est aléatoire et peut suivre différentes distributions : constante, normale, uniforme, orthogonale... L'importance de l'initialisation peut être diminuée notamment grâce à l'utilisation d'une normalisation des sous-ensembles d'apprentissage (IOFFE et SZEGEDY 2015).

Les couches de neurones

Une couche de neurones est un ensemble de neurones utilisant les mêmes entrées. L'ensemble de leurs sorties forment la sortie de la couche. Cette sortie est une projection dans un espace dont la dimension est le nombre de neurones de la couche.

Calcul effectué par une couche de S neurones :

$$a = f(Wp - b) \tag{2.10}$$

avec, pour une entrée de taille R :
$a : (S, 1), W : (S, R), p : (R, 1), b : (S, 1)$.

Il existe plusieurs types de couches de neurones, parmi lesquelles nous avons utilisé :

— **Dense** (GARDNER et DORLING 1998) : couche totalement connectée, illustrée sur la figure 2.25, dont chaque neurone utilise toute l'entrée. Les neurones ont donc le même nombre de poids que la taille de l'entrée.

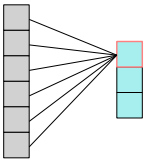

FIGURE 2.25 – Illustration d'une couche dense avec en noir les liens entre les entrées et un neurone de la couche

— **Convolution** (LECUN, BENGIO et al. 1995) : couche composée de filtres, illustrée sur la figure 2.26. Ce sont des neurones qui regardent l'entrée morceaux par morceaux (l'entrée doit être similaire à une image : par exemple un spectrogramme). La sortie d'une couche de convolution est environ de la taille de l'entrée (selon les paramètres de gestion des bords choisis) mais a une troisième dimension dont la taille est le nombre de neurones.

— **Bottleneck** (KRAMER 1991) : une couche plus petite que celle qui la précède et celle qui la suit, illustrée sur la figure 2.27. Cette couche peut servir par exemple à générer de nouvelles représentations paramétriques.

Les architectures de réseaux

La forme de base d'une réseau de neurones est le perceptron multi-couches (Multi-Layer Perceptron, MLP), illustré figure 2.28. C'est un réseau constitué de couches

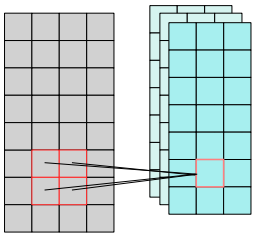

FIGURE 2.26 – Illustration de la sortie d'une couche de convolution de trois filtres de taille (2x2)

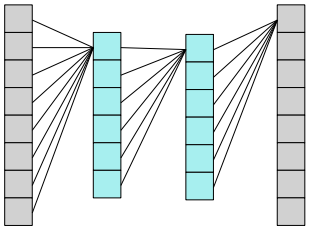

FIGURE 2.27 – Illustration d'une couche de Bottleneck

denses se suivant linéairement, dont chaque couche utilise en entrée la sortie de la couche précédente. Lorsqu'un réseau commence par une ou plusieurs couches de convolution, il s'appelle alors réseau convolutionnel (Convolutional Neural Network, CNN).

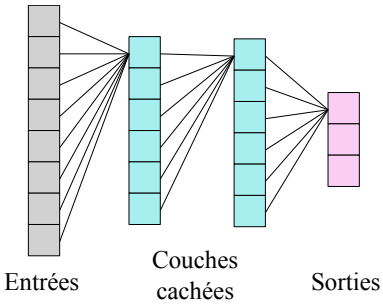

FIGURE 2.28 – Illustration d'un MLP avec l'entrée, la sortie, et les couches cachées entre

D'autres structures de réseaux existent, telles que les denseNet (G. HUANG et al. 2017). Dans ces réseaux, les relations entre les couches ne sont pas linéaires, chacune peut prendre les sorties de plusieurs couches en entrée, comme illustré dans la figure 2.29.

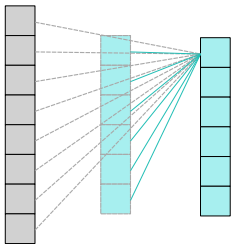

FIGURE 2.29 – Exemple d'une couche utilisant les entrées de plusieurs couches précédentes

L'apprentissage

L'apprentissage des poids des réseaux de neurones se fait en minimisant les coûts. Dans cette thèse, nous avons utilisée la rétro-propagation du gradient (voir algorithme 5) et l'erreur de reconstruction comme coût en non supervisé et l'entropie croisée comme coût en supervisé.

Algorithm 5 Exemple d'algorithme de rétropropagation du gradient

$w_{ij} \leftarrow$ initialisation aléatoire des poids
Tant que (condition), pour $(x, c) \in S$
. $o \leftarrow$ la sortie du réseau pour (x, c)
. Pour chaque sortie o_i de o
. $\quad \delta i \leftarrow o_i * (1 - o_i) * (c_i - o_i)$
. Fin Pour
. Pour chaque couche de $q - 1$ à 1
. $\quad$ Pour chaque cellule i de la couche
. $\quad\quad \delta i \leftarrow o_i(1 - o_i) * [\sum(k \in Succ(i))(\delta k * w_{ki})]$
. $\quad$ Fin Pour
. Fin Pour
. Pour chaque poids w_{ij}
. $\quad w_{ij} \leftarrow w_{ij} + \epsilon * \sigma_i * x_{ij}$
. Fin Pour
Fin Tant que

Dans le cas d'apprentissage par descente du gradient des optimisations des mises à jours peuvent être utilisées (RUDER 2016), comme le momentum (N. QIAN 1999) utilisant la dérivée, ou d'autres méthodes adaptent le taux d'apprentissage en utilisant la racine carrée des gradients cumulés (Adagrad) ou encore la moyenne des racines carrées des gradients (rmsprop). Plus précisément :

- **Stochastic Gradient Descent (SGD)**. Mise à jour basique à l'aide du gradient multiplié par le taux d'apprentissage.
- **Momentum SGD** (N. QIAN 1999). Mise à jour accélérée dans la direction principale des dernières mises à jour à l'aide de la dérivée.
- **Adagrad** (DUCHI, HAZAN et SINGER 2011). Mise à jour différente pour chaque paramètre : plus importante pour les moins fréquents et plus faible pour les plus fréquents.
- **Adadelta** (ZEILER 2012). Variante de l'Adagrad, avec un historique limité pour évaluer la fréquence des paramètres.
- **RMSprop** [18]. Développé parallèlement à l'Adadelta, RMSprop est lui aussi une variante de l'Adadelta, utilisant la racine carrée pour limiter la diminution du taux d'apprentissage des paramètres concernés.
- **Adam** (KINGMA et BA 2015). Mise à jour des poids adaptatif comme l'Adagrad, corrigé comme l'Adadelta et le RSMprop, et prenant en compte l'axe dominant de l'historique (la dérivée des gradients), comme le Momentum.
- **Adamax** (KINGMA et BA 2015). Un des hyperparamètres d'Adam est fixé selon une échelle inversement proportionnelle à la norme l2 des gradients passés et du gradient présent.

Il faut faire attention au sur-apprentissage. En effet, ceci apparaît lorsque le taux de classification de l'ensemble de test diminue alors qu'il peut se rapprocher de 100% sur l'ensemble d'apprentissage (avec suffisamment de filtres et de neurones).

18. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Les librairies Python

À l'heure actuelle, il existe de nombreuses bibliothèques haut niveau, par exemple Keras [19], PDNN (avec Theano et Kaldi) (MIAO 2014), PyBrain (SCHAUL et al. 2010), Tensorflow (ABADI et al. 2016), ... Pour aider à choisir, il existe des articles de synthèse tel que (ERICKSON et al. 2017).

Dans cette thèse, nous avons utilisé Theano (BASTIEN et al. 2012 ; TEAM 2016) et Lasagne (DIELEMAN et al. 2015) pour la mise en œuvre des réseaux de neurones. Lasagne est une bibliothèque assez haut niveau, permettant de coder des réseaux en peu de lignes, et Theano était l'outil le plus stable au début de cette thèse.

2.3 Corpora

Le corpus le plus utilisé en traitement de la parole est probablement TIMIT (GAROFOLO 1993), un corpus d'anglais lu, mais ce n'est pas ce corpus que nous avons choisi d'utiliser. En effet, nous avons préféré effectuer le plus gros de nos recherches sur un corpus d'anglais conversationnel, BUCKEYE, utilisé lors du challenge *Zero Resource Speech* de 2015.

En tout, plusieurs corpora ont été utilisés durant les recherches effectuées dans le cadre de cette thèse :

— **BUCKEYE** (M. PITT et al. 2007) : anglais américain, parole conversationnelle,
— **NCHLT** (HEERDEN, M. H. DAVEL et BARNARD 2013) : xitsonga (langue d'Afrique du sud), parole lue,
— **BREF** (L. F. LAMEL et al. 1993) : français, parole lue,
— **Corpora du *ZRSC 2017*** (DUNBAR et al. 2017)
  • 3 langues de développement : anglais, français et mandarin, parole lue,
  • 2 langues de test « surprise » : L1 et L2, parole lue

2.3.1 BUCKEYE

BUCKEYE est le corpus que nous avons le plus utilisé. Il est constitué d'enregistrements de parole conversationnelle provenant d'une quarantaine de locuteurs différents (hommes, femmes, jeunes et moins jeunes) américains natifs avec pour chacun entre 30 minutes et 1 heure de durée d'enregistrement. Ce corpus est décrit en détails dans (KIESLING, DILLEY et RAYMOND 2006). Il contient un peu plus de 100 000 réalisations d'environ 10 000 mots différents, dont la majorité ont moins de trois syllabes et souvent une seule. Il utilise 59 annotations de phonèmes différents, et la durée médiane des phonèmes est de 70 ms. Le nombre de phonèmes est supérieur à la quarantaine habituellement référencé en anglais, notamment à cause de prononciations particulières que les auteurs de BUCKEYE ont choisi d'isoler dans des classes différentes, pour les nasales en particulier. Celles-ci sont différenciées par une annotation se terminant par "n", comme nous pouvons le voir dans la table **??** où sont rapportées les différentes notations utilisées par les créateurs du corpus BUCKEYE. Dans certains graphiques de cette thèse, leurs annotations ont été privilégiées aux écritures phonétiques.

La qualité des annotations manuelles a été évaluée par les créateurs du corpus. Un accord inter-annotateurs a été calculé : il est de l'ordre de 62% de F-mesure pour la segmentation avec 10 ms de marge (décalage). Le pourcentage monte à 79% pour

19. https://keras.io/

Voyelles

Buckeye	IPA
aa	ɑ
aan	ɑ̃
ae	æ
aen	æ̃
ah	ə / ʌ
ahn	ə̃ / ʌ̃
ao	ɔ
aon	ɔ̃
aw	aʊ
awn	aʊ̃
ay	aɪ
ayn	aɪ̃
eh	ɛ
ehn	ɛ̃
el	l̩
em	m̩
en	n̩

Buckeye	IPA
er	ɹ̩
ern	ɹ̩̃
ey	eɪ
eyn	eɪ̃
ih	ɪ
ihn	ɪ̃
iy	i
iyn	ĩ
ow	oʊ
own	oʊ̃
oy	ɔɪ
oyn	ɔɪ̃
uh	ʊ
uhn	ʊ̃
uw	u
uwn	ũ

Consonnes

Buckeye	IPA
b	b
ch	ʧ
d	d
dh	ð
dx	ɾ
f	f
g	g
hh	h
jh	ʤ
k	k
l	l / ɬ
m	m
n	n

Buckeye	IPA
ng	ŋ
p	p
r	ɹ
s	s
sh	ʃ
t	t
th	θ
tq	ʔ
v	v
w	w
y	j
z	z
zh	ʒ

FIGURE 2.30 – Annotations utilisées par les auteurs du corpus BUCKEYE pour les voyelles et les consonnes, avec les voyelles nasales différenciées par une écriture se terminant par "n"

un décalage de 20 ms. Le score inter-annotateur de la transcription phonétique se situe entre 76 et 80%. Les annotateurs assignent le même label dans 62% des cas (RAYMOND et al. 2002 ; M. A. PITT et al. 2005).

2.3.2 BREF

BREF est un corpus de parole lue en français (L. F. LAMEL et al. 1993). Comme nous nous intéressons aux langues à faibles ressources, nous n'avons pris qu'une heure de parole pour nos expériences, issue du sous-corpus BREF80 et enregistrée par 8 locuteurs différents, avec un peu moins de 10 minutes pour chaque. L'ensemble phonétique français considéré est l'ensemble standard, composé de 35 phonèmes différents, dont la durée médiane est de 70 ms.

2.3.3 NCHLT

Le corpus en langue xitsonga (HEERDEN, M. H. DAVEL et BARNARD 2013) est composé de courtes phrases lues, enregistrées sur smartphone hors studio. Nous avons utilisé environ 500 phrases, avec en tout 10 000 exemples de phonèmes. La durée médiane est d'environ 90 ms et il y a 49 phonèmes différents.

2.3.4 Corpora du ZRSC 2017

Tous les ensembles de test du ZRSC 2017 sont découpés en 3 sous-ensembles, regroupant les fichiers de 1s, 10s et 120s. Les ensembles de développement (pour les trois langues connues) ont en plus des ensembles dits d'apprentissage, non évalués

et de tailles différentes (45 heures, 24 heures et 2 heures 30 respectivement), que nous n'avons pas utilisés.

La table 2.4 affiche les durées d'enregistrement de ces langues. Nous avons utilisé 27 heures d'anglais, 17 heures de français et 25 heures de mandarin.

TABLE 2.4 – Durée des corpus de test du Zero Resource Speech Challenge 2017

Langage	Anglais	Français	Mandarin	L1	L2
Durée (heures)	27	17	25	11	6

2.3.5 Étude comparative de BUCKEYE, NCHLT et BREF

BUCKEYE étant le corpus principalement utilisé durant cette thèse, nous avons étudié sa composition (répartition des phonèmes, variabilité inter-locuteurs…) et, occasionnellement, comparé avec le corpus NCHLT de xitsonga lu.

La figure 2.31 montre un exemple de spectrogramme issus de BUCKEYE (avant puis après l'avoir centré/normé) et, pour comparer, un exemple issu de NCHLT. Les deux graphiques du bas (moyennes/variances des bancs de filtres issus du corpus d'anglais et de xitsonga) permettent plus facilement d'analyser les différences des spectrogrammes. Ainsi, nous voyons que le xitsonga utilise davantage les hautes fréquences, peut-être à cause des consonnes occlusives orales (le xitsonga est dit une langue à « clic » à cause de la forte présence de consonnes claquantes). En revanche, sa variabilité dans les hautes fréquences n'est pas plus élevée que dans les basses fréquences, contrairement à l'anglais. Les hautes fréquences semblent donc autant présentes que les autres fréquences.

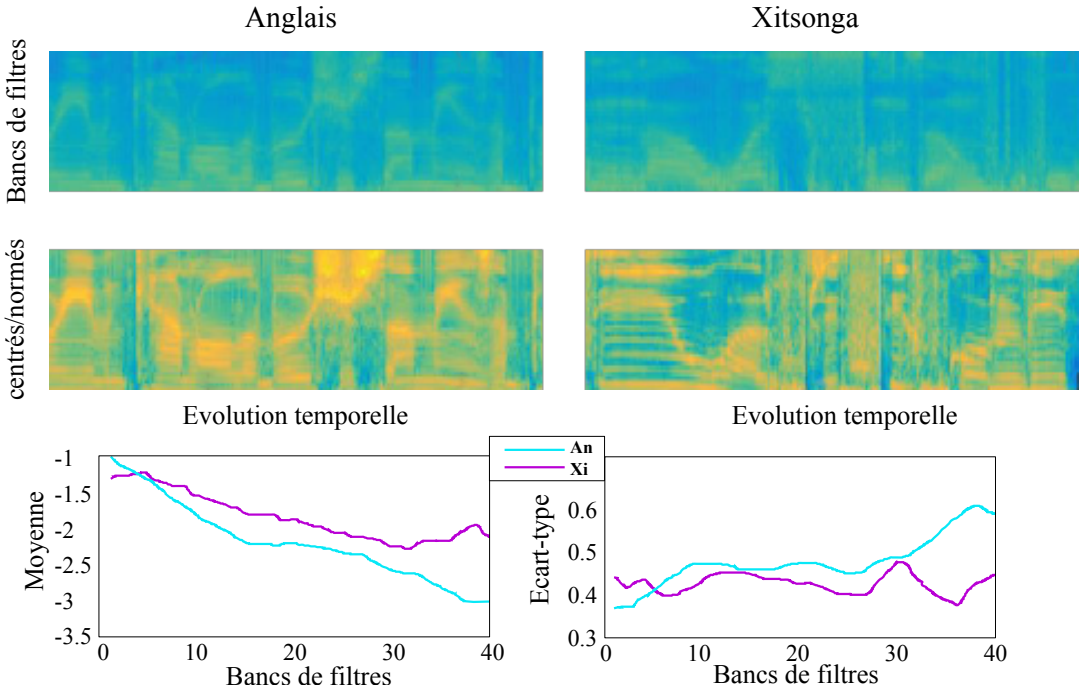

FIGURE 2.31 – Extraits de spectrogrammes (en haut) de bancs de filtres pour des extraits d'anglais et de xitsonga normés et non normés d'une durée d'environ 1s et comparaison moyenne/variance de l'ensemble des corpus (en bas)

La figure 2.32 montre la variabilité inter-locuteur de la moyenne et de l'écart-type des bancs de filtres : chaque ligne correspond à un locuteur différent. Nous voyons une assez bonne homogénéité des moyennes et davantage de différence dans les variances, principalement dans les hautes fréquences.

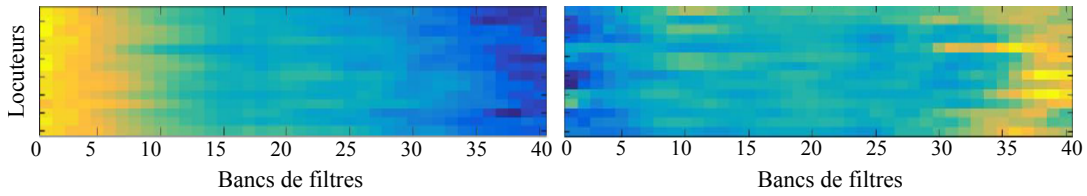

FIGURE 2.32 – Moyennes (*à gauche*) et écarts-types (*à droite*) des bancs de filtres pour différents locuteurs dans BUCKEYE

La figure 2.33 représente l'histogramme du nombre d'occurrences de chaque phonème dans le corpus BUCKEYE et le sous-ensemble d'une heure issu de BREF. Nous voyons qu'il y a beaucoup de phonèmes faiblement représentés dans BUCKEYE, contrairement à BREF qui est un peu plus homogène. C'est probablement dû aux plus nombreuses écritures phonétiques dans BUCKEYE : 59 contre 35 pour BREF. Les trois phonèmes de BUCKEYE les plus présents se démarquant sur le graphique sont le ə, le ɪ et le s.

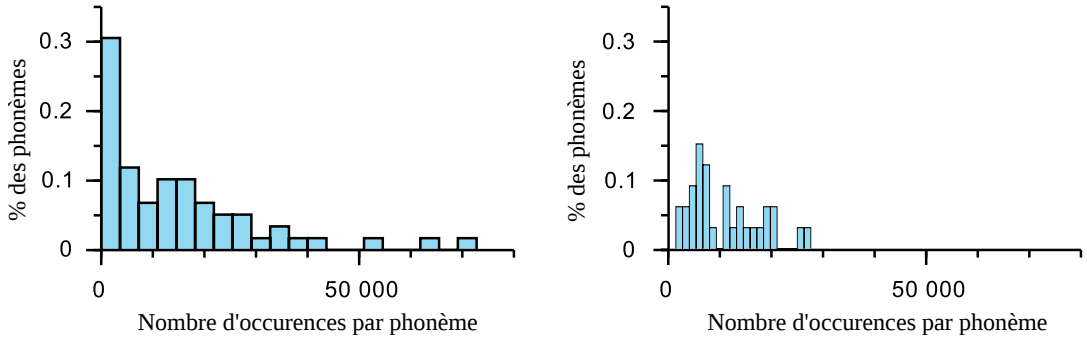

FIGURE 2.33 – Histogramme du nombre d'occurrences des phonèmes présents dans BUCKEYE (*à gauche*) et dans le sous-ensemble de BREF80 (*à droite*)

L'histogramme 2.34 montre la répartition des différentes tailles des segments phonétiques. La durée médiane des phonèmes est d'environ 70 ms pour les deux corpora, bien que le français soit de la parole lue et l'anglais de la parole conversationnelle.

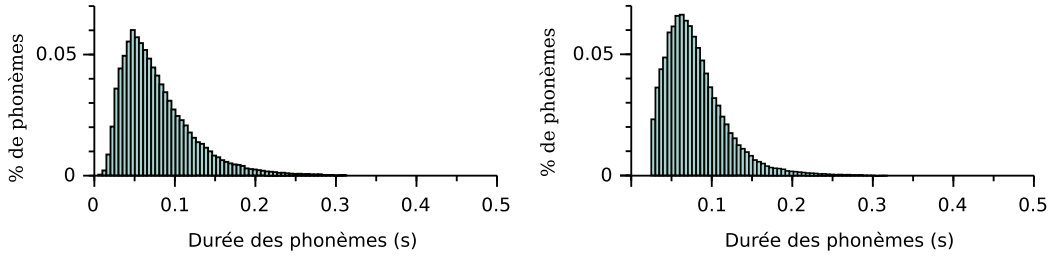

FIGURE 2.34 – Histogramme des durées des réalisations des phonèmes dans BUCKEYE (*à gauche*) et le sous-ensemble de BREF80 *(à droite*)

Généralement, les travaux effectués sur les phonèmes utilisent le signal de la parole en trames de 25 ms, or dans BUCKEYE 0,3% des phonèmes ont une taille inférieure, ce qui rendra ces segments difficiles à identifier. Au contraire, dans le sous-ensemble d'une heure du corpus de parole lue BREF, seulement 6 réalisations ont une taille inférieure à 25 ms. Ainsi, bien que les phonèmes des deux corpus aient la même durée médiane, leur répartition est plus homogène dans le corpus de parole lue puisqu'ils ont beaucoup moins de phonèmes très courts.

2.4 Conclusion

L'état de l'art nous a permis de valider notre choix d'utiliser des réseaux de neurones durant cette thèse, de nous renseigner sur les méthodes déjà existantes et les paramètres souvent utilisés en traitement automatique de la parole.

Nous avons ensuite approfondi notre connaissance des principaux outils (paramètres, méthodes, bibliothèques et corpora) utilisés dans la suite de notre travail.

Chapitre 3

Classification phonétique supervisée à l'aide de réseaux de neurones

Sommaire

3.1 Introduction

Comme nous l'avons vu dans l'état de l'art (Chapitre 2), les réseaux de neurones ont une place de plus en plus importante en traitement automatique de la parole. Dans ce chapitre, nous allons identifier des architectures de réseaux pertinentes ainsi que fixer les divers paramètres de manière supervisée. Cela nous permettra de mieux appréhender le cas non-supervisé qui est au cœur de cette thèse.

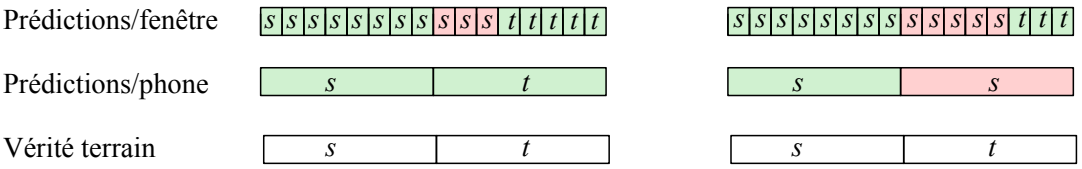

FIGURE 3.1 – Illustrations des deux mesures du taux de classification (au niveau des fenêtres et au niveau des segments phonétiques

Afin d'optimiser l'architecture de nos modèles, nous évaluons les résultats à l'aide du taux de classification correcte des phonèmes. Deux niveaux sont considérés (*cf.* figure 3.1) :

— le niveau fenêtre ou trame d'analyse acoustique, désigné par *taux_fen*. Il correspond au pourcentage de fenêtres correctement classées ;
— le niveau segment phonétique, désigné par *taux_seg*. Il tient compte des frontières de phonèmes de la vérité terrain. L'attribution d'un phonème à un segment est réalisée à l'aide d'un vote majoritaire sur les fenêtres composant le segment.

Dans le premier exemple de la figure (à gauche), le taux au niveau des segments phonétiques est meilleur (100%) que celui au niveau des fenêtres et c'est l'inverse dans le deuxième exemple (à droite)

Toutes les expériences de ce chapitre ont été effectuées sur le corpus d'anglais conversationnel BUCKEYE, constitué d'une quarantaine d'heures de parole avec une quarantaine de locuteurs différents. Ce corpus est divisé en trois ensembles d'environ 20, 10 et 10 heures pour BUCKEYE-TRAIN, BUCKEYE-DEV et BUCKEYE-TEST, respectivement, avec les locuteurs présents dans les trois ensembles. Pour plus de détails sur le corpus, voir Section 2.3.

Dans ce chapitre, nous allons tout d'abord décrire des expériences visant à construire des réseaux de neurones efficaces et optimiser la valeur des (hyper)paramètres. Nous étudierons ensuite les résultats obtenus, les erreurs les plus fréquentes puis la robustesse et la portabilité du réseau choisi avant de conclure.

3.2 Architectures et ajustement des paramètres

Les réseaux de neurones sont connus pour s'adapter plus facilement que d'autres méthodes à des problèmes très divers, mais ils ont eux aussi des paramètres à optimiser. Pour ajuster ces paramètres, qui définissent aussi bien la structure du réseau (nombre de couches, paramètres d'entrée, ...) que son apprentissage (taux d'apprentissage, règle de mise à jour des poids, ...), nous avons réalisé diverses expériences dont nous avons reporté les résultats dans cette section. Deux architectures ont été envisagées : les réseaux denses (MLP) et les réseaux convolutionnels (CNN).

3.2.1 Paramètres classiques pour l'apprentissage d'un réseau de neurones

Nous avons utilisé l'algorithme de rétropropagation du gradient de la fonction de coût pour entraîner les réseaux. Les paramètres standard à optimiser dans ce cadre sont les suivants :

— **La taille des sous-ensembles (mini-batchs).** Les données sont utilisées par petits sous-ensembles (souvent choisis aléatoirement) lors de l'apprentissage afin de mettre à jour les paramètres du réseau.
— **La règle de mise à jour des poids.** Divers algorithmes d'optimisation de la rétro-propagation du gradient existent dans le but d'améliorer la convergence.
— **Le taux d'apprentissage.** Il permet de converger plus ou moins rapidement vers une solution. Le choix de ce paramètre est délicat : s'il est trop petit alors la convergence peut être trop lente ; s'il est trop grand alors des oscillations de la fonction coût peuvent apparaître, et il peut même ne pas converger…
— **La régularisation des poids**. En particulier, le dropout srivastava2014dropout consiste à éteindre certains neurones lors de l'apprentissage pour améliorer les capacités de généralisation du réseau et ainsi lutter contre le sur-apprentissage.

Influence de la taille des sous-ensembles

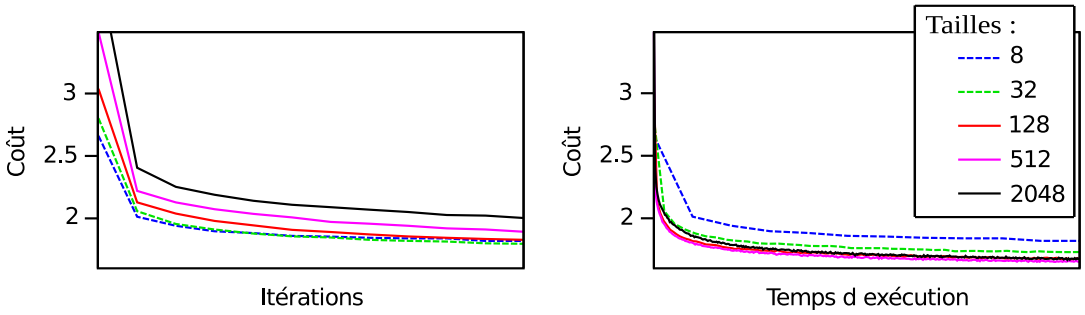

FIGURE 3.2 – Influence de la taille des sous-ensembles d'apprentissage sur la diminution du coût selon le nombre d'itérations (*graphique de gauche*) et le temps d'exécution (*graphique de droite*)

Les données d'apprentissage ne sont pas toutes utilisées simultanément mais par sous-ensemble. Ainsi, la correction des poids des neurones est faite à l'aide des erreurs de prédiction du réseau de neurones et chacune des corrections est effectuée à partir des prédictions faites sur un sous-ensemble composé de plusieurs exemples, de 1 à 2048 exemples dans nos expériences.

Cette taille est majorée par les capacités de la carte graphique (GPU) utilisée et doit être suffisante pour les calculs d'optimisation de mise à jour des poids. Durant nos expériences, la taille des sous-ensembles a eu peu d'influence sur les résultats. Pour évaluer l'impact de ce paramètre, nous avons comparé les temps d'exécution et l'évolution de la fonction de coût en utilisant le locuteur *s01* avec un modèle dense.

Nous avons mesuré les durées d'apprentissage et reporté les résultats dans les graphiques de la figure 3.2. Les courbes représentent l'évolution du coût pour différentes tailles de sous-ensembles en fonction du nombre d'itérations (graphique de gauche) ou en fonction du temps d'exécution (graphique de droite). La taille choisie a des conséquences sur le temps de calcul et l'efficacité de chaque itération. Plus la taille des sous-ensembles est grande, plus il faut d'itérations pour atteindre un

même coût (voir graphique de gauche). Cependant, en considérant le temps d'exécution, le coût est quasiment identique pour les différentes tailles (voir graphique de droite) sauf pour des tailles de sous-ensembles trop petites (taille 8 et 32).

En conclusion, la taille des sous-ensembles n'a pas beaucoup d'influence sur la durée et la qualité de l'apprentissage dans l'exemple considéré. D'autres expériences avec des paramètres différents (règle de mise à jour des poids et taux d'apprentissage) ont donnés des résultats légèrement différents, montrant une dépendance entre ces paramètres. Par la suite, nous avons choisi d'utiliser une taille de sous-ensembles de 512 exemples, qui dans cet exemple minimise le coût pour une même durée d'apprentissage.

Influence de la règle de mise à jour des poids

Les règles de mise à jour des poids que nous avons testées sont : SGD, Nesterov Momentum, Rmsprop, Adadelta, Adam, Adamax. Celles-ci ont été définies dans le chapitre 2.

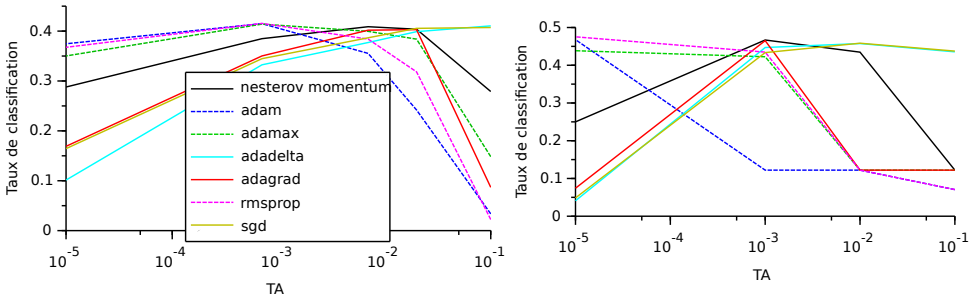

FIGURE 3.3 – Résultats selon le taux d'apprentissage et la règle de mise à jour des poids choisie pour un réseau dense (*à gauche*) et un réseau convolutionnel (*à droite*)

Selon les cas (type du réseau, nature des données d'apprentissage), ce n'est pas toujours la même règle qui obtient les meilleurs résultats. Dans nos expériences, plusieurs règles aboutissent à des résultats similaires. Nous voyons dans la figure 3.3 que pour BUCKEYE, avec un modèle dense, les fonctions obtiennent des résultats similaires mais avec un taux d'apprentissage différent. Il est donc préférable de fixer une règle de mise à jour et ensuite d'optimiser ses paramètres. Par la suite, nous utilisons Adamax (pour les réseaux denses) ou Nesterov Momentum (pour les réseaux convolutionnels), qui ont de bons résultats sur les plus larges plages de valeurs, facilitant l'ajustement du taux d'apprentissage.

Chacune de ces règles a des paramètres à ajuster pour améliorer les résultats. Le seul paramètre commun à toutes est le taux d'apprentissage que nous avons cherché à optimiser dans le paragraphe suivant. Les autres paramètres sont ceux utilisés par défaut dans les implémentations de Lasagne lasagne.

Influence du taux d'apprentissage

Comme nous allons le voir dans cette section, de même que dans d'autres expériences rapportées plus loin, le choix du taux d'apprentissage a davantage d'impact sur les résultats que les précédents paramètres étudiés. Si sa valeur est trop élevée ou trop faible, les résultats obtenus sont proches du bruit. Les différences obtenues selon sa valeur sont visibles en regardant les poids des filtres de la première couche de convolution illustrés dans la figure 3.4.

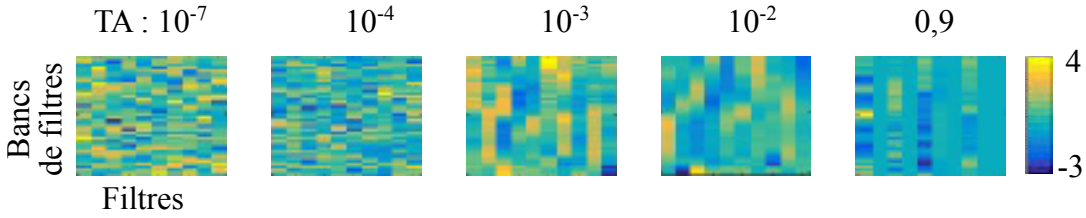

FIGURE 3.4 – Valeurs des poids des filtres de la première couche de convolution selon le taux d'apprentissage (TA), allant de trop faible (*à gauche*) à trop élevé (*à droite*)

La figure 3.4 montre des exemples de valeurs prises par des filtres d'une première couche de convolution apprise sur des bancs de filtres de taille 40 (nombre de paramètres) selon le taux d'apprentissage. La conséquence d'un taux d'apprentissage trop élevé est la présence de beaucoup de filtres nuls et celle d'un taux d'apprentissage trop faible est d'obtenir des filtres proches du bruit. Dans cet exemple, issu du locuteur *s01*, nous voyons que le taux d'apprentissage idéal se trouve aux alentours de 10^{-2} et 10^{-3}, pour lesquels les filtres ressemblent le plus à des exemples de bancs de filtres. Nous avons conservé cette plage de valeurs pour les expériences décrites dans la suite du manuscrit.

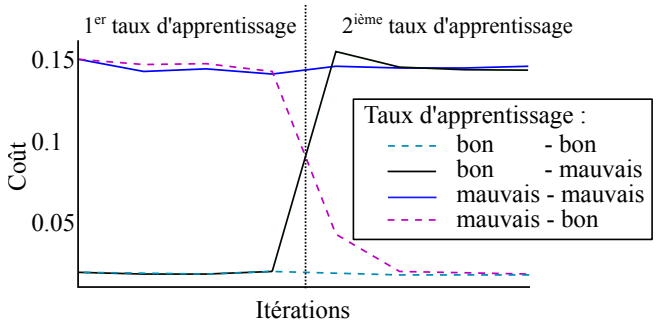

FIGURE 3.5 – Évolution du coût selon le taux d'apprentissage : bon (proche du taux idéal) ou mauvais (trop élevé ou trop faible). Le taux d'apprentissage est modifié en milieu d'apprentissage

Il existe différentes méthodes permettant d'optimiser cet important paramètre. Par exemple, jacobs1988increased conseillait d'utiliser un taux d'apprentissage variant dans le temps et différent pour chaque poids du réseau, ce que font maintenant beaucoup de méthodes d'optimisation de mise à jour des poids, comme nous l'avons vu rapidement chapitre 2.

En plus de la valeur initiale du TA, il peut être intéressant de mettre en œuvre une stratégie de changement de sa valeur au cours de l'apprentissage, comme par exemple une diminution en fin d'apprentissage. Les règles de mise à jour des poids les plus récentes telles que Adamax zeiler2012adadelta et Adam kingma2014adam facilitent en théorie le choix d'une telle stratégie grâce à leur adaptation dynamique du TA.

La figure 3.5 montre l'évolution du coût selon le TA. Nous voyons que le plus important est d'avoir le bon taux en fin d'apprentissage. Lorsque nous commençons à appréhender un nouveau problème (nouveau réseau, nouvelles données, nouvelle tâche), partir d'un taux faible et augmenter jusqu'à un taux efficace peut être la solution la plus rapide pour trouver le taux idéal.

Influence du pré-apprentissage couche par couche

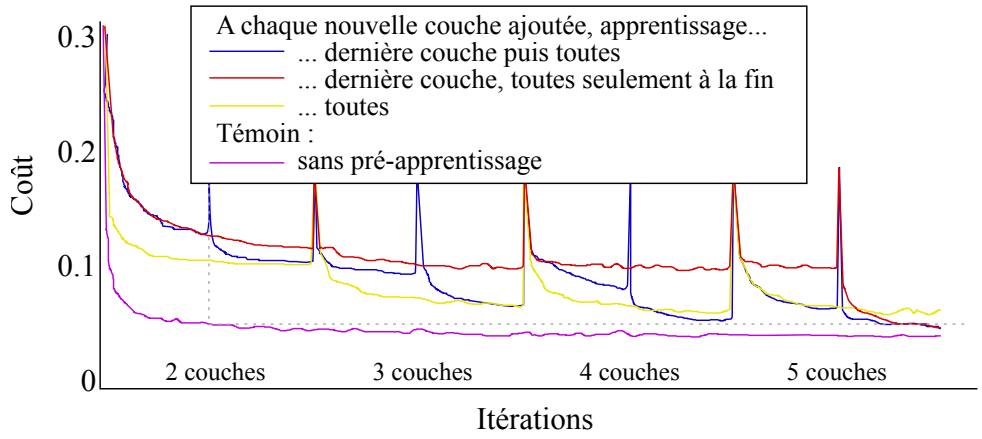

FIGURE 3.6 – Influence du pré-apprentissage couche par couche sur l'apprentissage d'un MLP pour la classification de phonèmes

Plus un réseau est profond, plus son entraînement est long et difficile. Différentes méthodes sont apparues pour essayer de remédier à ce problème. Parmi ces méthodes, nous avons testé la construction et l'apprentissage couche par couche du réseau knerr1990single,bengio2007greedy. La figure 3.6 montre l'évolution du coût avec ou sans pré-entraînement. Différentes approches de pré-entraînement sont illustrées : en figeant ou pas les couches précédant une nouvelle couche, en entraînant ou pas tout le réseau après l'ajout d'une couche (*fine-tuning*). Les pics que nous voyons sur la figure sont une augmentation brutale du coût, survenant à chaque ajout d'une nouvelle couche. Ils peuvent être dus à un temps d'adaptation aux nouveaux paramètres nécessaires pour la règle d'optimisation utilisée. Ces changements semblent n'avoir d'impact que sur les premières itérations mais pas sur la performance finale du modèle.

Les différentes méthodes obtiennent majoritairement des résultats similaires :

— **Apprentissage de tout le réseau à chaque nouvelle couche ajoutée** (en jaune sur la figure) : à chaque nouvelle couche ajoutée, le coût diminue plus rapidement que les deux autres approches, mais stagne ensuite et au final il obtient les moins bons résultats. Peut-être est-il utile d'adapter la dernière couche ajoutée aux sorties de la couches précédentes avant de faire remonter ses erreurs dans les couches déjà apprises

— **Apprentissage de la dernière couche ajoutée et de tout le réseau seulement à la fin** (en rouge sur la figure) : le coût ne diminue que très peu en ne réapprenant pas le reste du réseau, mais une fois cet apprentissage fait (seconde moitié de la dernière partie du graphique, indiquée '5 couches') les résultats rattrapent ceux des autres approches et sont même meilleurs que l'approche précédente. Il semble donc que cette approche permettent d'obtenir de bons résultats mais que peu d'itérations sont utiles pour l'apprentissage de chaque nouvelle couche. Son autre avantage par rapport à la méthode précédente est que son apprentissage est plus rapide, puisqu'il est plus rapide d'apprendre une seule couche que plusieurs

— **Apprentissage de la dernière couche ajoutée puis de tout le réseau déjà construit** (en bleu sur la figure) : obtient au final les mêmes résultats que la méthode précédente, mais avec de meilleurs résultats durant l'apprentissage des couches intermédiaires, permettant potentiellement de pouvoir fixer

la taille du réseau pendant l'apprentissage en arrêtant lorsque l'ajout d'une couche n'est plus bénéfique aux résultats. Le dernier apprentissage global est plus rapide que celui de la méthode précédente, le réseau étant déjà davantage appris

— **Sans pré-apprentissage** (en violet sur la figure) : les résultats sont meilleurs pour un même nombre d'itérations, mais les itérations sont plus longues puisque tout le réseau final est appris à chaque fois et non pas seulement quelques couches. Pour un nombre d'itérations avec tout le réseau égal aux autres, ses résultats sont similaires (pointillés gris)

Des résultats contraires ont été obtenus dans la littérature en particulier si la tâche d'apprentissage s'avère difficile. Par exemple, kamper2017unsupervised souligne que le pré-entraînement des couches de ses auto-encodeurs a été nécessaire pour obtenir de bons résultats en découverte non supervisée de pseudo-phones.

Influence de la régularisation par dropout

Utiliser du dropout (ne pas utiliser tous les neurones lors de l'apprentissage) est utile pour augmenter la robustesse en forçant le réseau, pour remédier aux disparitions aléatoires de ses neurones, à ajouter de la redondance dans les calculs.

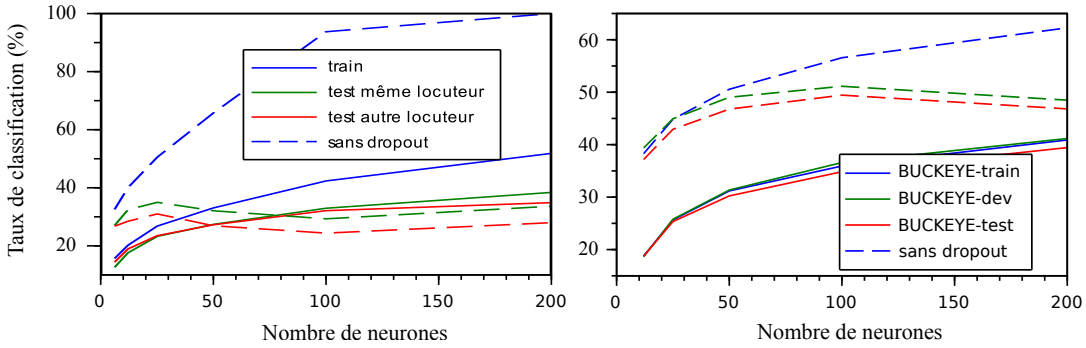

FIGURE 3.7 – Influence du dropout sur la robustesse à de nouvelles données d'un MLP pour la classification de phonèmes pour un unique locuteur dans l'ensemble d'apprentissage (*à gauche*) ou tout le corpus d'apprentissage (*à droite*)

Pour tester l'utilité de l'ajout de dropout sur les couches (à la valeur par défaut de 50%), nous avons comparé les taux de classification obtenus sur l'ensemble d'apprentissage (pour un seul locuteur), un ensemble de test issu du même locuteur et un ensemble de test contenant le reste du corpus BUCKEYE. Nous voyons sur la figure 3.7 que l'ajout du dropout nécessite un réseau avec beaucoup plus de neurones pour obtenir un résultats similaires à un réseau sans dropout. Sur le graphique de gauche (peu d'apprentissage : seulement 1 locuteur), il faut environ 100 neurones par couches avec dropout pour obtenir un résultat similaire à des couches de 25 neurones sans dropout. Sur le graphique de droite, où tout le corpus BUCKEYE est utilisé, le réseau sans dropout obtient de meilleurs résultats qu'avec dropout pour moins de 200 neurones par couches. Nous constatons que cette régularisation permet de réduire de manière importante le sur-apprentissage (sans cela, le réseau obtient en effet un taux de classification avoisinant les 100% sur l'ensemble d'apprentissage pour plus de 100 neurones par couche sur le graphique de gauche). Néanmoins, notre but étant de nous diriger vers un modèle non supervisé, nous choisissons de

ne pas utiliser de dropout par la suite pour réduire la taille du réseau, car plus il y a de poids à apprendre plus il y a besoin de données pour les entraîner.

3.2.2 Paramètres propres à un réseau de neurones dense

Pour la construction d'un réseau de neurones dense (Multi-Layer Perceptron, MLP), les paramètres que nous devons choisir sont :

— **Les paramètres d'entrée** (paramètres acoustiques et taille du voisinage)
— **Le nombre de couches et de neurones par couche**
— **La fonction d'activation de chaque couche**

Influence des paramètres d'entrée

Les MFCC sont des paramètres efficaces avec les MLP. Nous avons effectivement pu constater au cours de nos expériences que d'autres paramètres n'étaient pas adaptés, tels que les bancs de filtres qui n'ont pas permis d'obtenir de bons résultats : environ 20% de moins que les MFCC. Par la suite, nous avons utilisé 12 MFCC statiques et l'énergie ainsi que les dérivées premières et secondes.

Considérer le contexte d'une trame acoustique est utile pour déterminer une classe phonétique associée. Nous avons donc étudié l'influence de la taille du voisinage en entrée des MLP.

Les unités de parole sont considérées stables sur plusieurs dizaines de millisecondes (autour de 50 ms), nous pouvons donc nous attendre à un voisinage d'au moins cette durée. Plus le voisinage considéré est grand, plus le réseau doit traiter un nombre important de données en entrée. Les résultats affichés dans la figure 3.8 nous permettent de choisir un voisinage de 18 fenêtres, soit 84 ms.

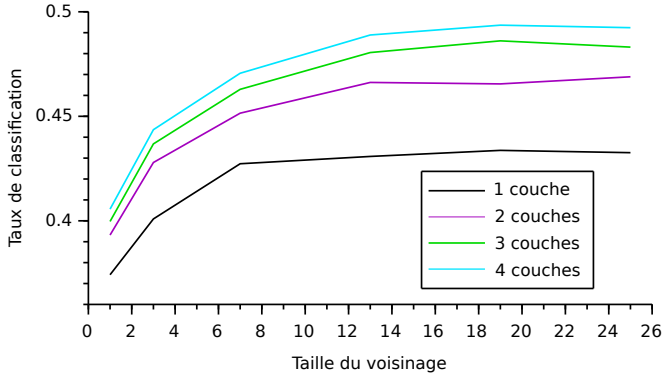

FIGURE 3.8 – Taux de classification obtenu selon la taille du voisinage utilisé en entrée pour différents réseaux (de 1 à 4 couches denses cachées)

Influence du nombre de couches et de neurones

Ici nous nous intéressons à la complexité des MLP nécessaire pour atteindre notre objectif. Plus un MLP va être profond, plus il aura d'expressivité. Cependant, comme dit précédemment, il sera d'autant plus difficile de les entraîner de manière satisfaisante.

Il n'existe pas de choix idéal unique du nombre de couches et du nombre de neurones pour un problème donné. Il est prouvé mathématiquement qu'un réseau avec

une seule couche cachée peut approximer n'importe quelle fonction mathématique. Cependant le nombre de neurones nécessaire peut être prohibitif en pratique. Utiliser plusieurs couches, et donc des réseaux profonds, permet d'atteindre le même objectif mais avec un nombre de neurones raisonnable.

Le graphique 3.9 montre les taux de classification obtenus sur BUCKEYE-DEV selon le nombre de couches et de neurones par couche. Les réseaux utilisant du dropout ont obtenu des résultats contraires à ce que nous pensions obtenir : les réseaux avec peu de couches ont été meilleurs que ceux avec beaucoup de couches dans le cas d'un nombre de neurones par couche petit (< 200). Sans dropout, nous avons effectivement obtenu ce que nous attendions : les réseaux les plus profonds préfèrent avoir moins de neurones que des réseaux d'une ou deux couches cachées.

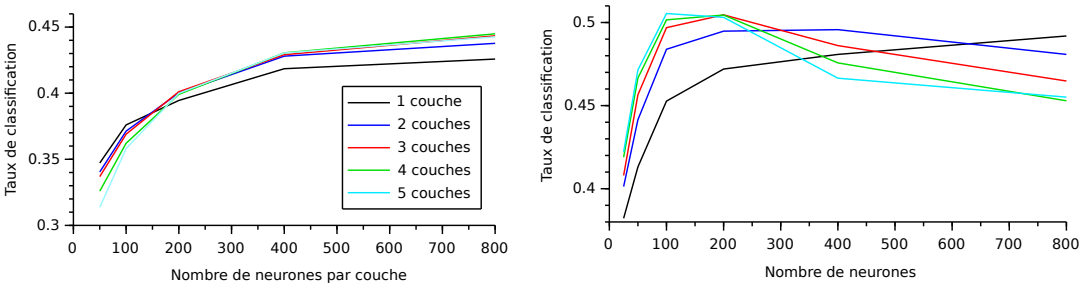

FIGURE 3.9 – Taux de classification obtenu sur BUCKEYE-DEV par un MLP en fonction du nombre de couches cachées et du nombre de neurones par couche, avec dropout (*à gauche*) ou sans (*à droite*)

Influence de la fonction d'activation

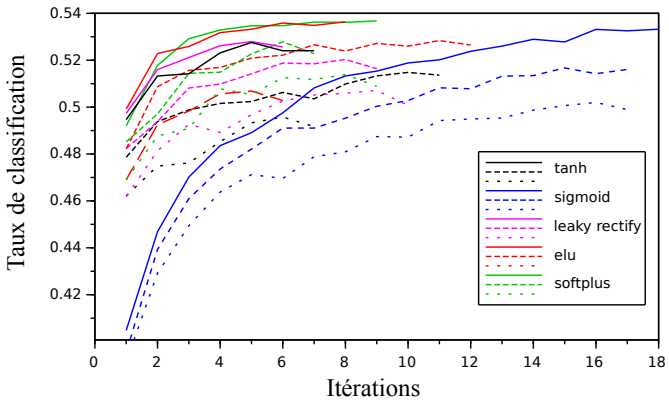

FIGURE 3.10 – Résultats selon la fonction d'activation des couches p our des réseaux de 2, 3 ou 4 couches cachées (resp. pointillés légers, pointillés et trait plein)

Nous avons testé 5 fonctions d'activations différentes : tangente hyperbolique, *sigmoïde*, leaky Rectified Linear Units (Leaky ReLU), Exponential Linear Units (ELU) et *softplus*. La figure 3.10 montre les résultats obtenus sur BUCKEYE-DEV avec un même réseau et un même taux d'apprentissage. Dans cette expérience, les meilleures fonctions sont *softplus* et *ELU*.

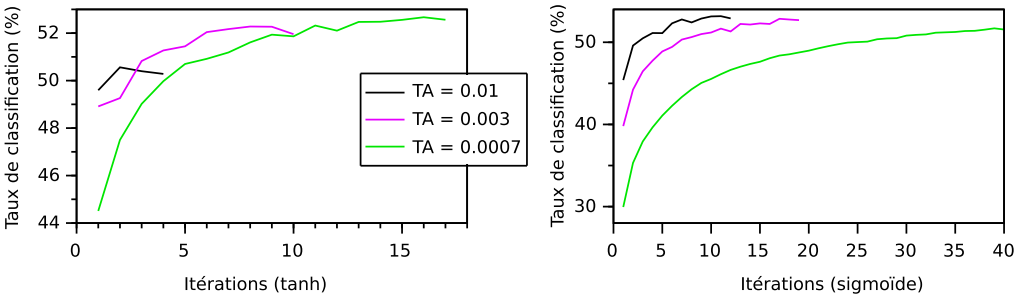

FIGURE 3.11 – Résultats selon le Taux d'Apprentissage (TA) pour la tangente hyperbolique (*à gauche*) et la fonction *sigmoïde* (*à droite*)

L'évolution lente du réseau utilisant la *sigmoïde* montre qu'il aurait fallu pour cette configuration un taux d'apprentissage plus élevé pour converger aussi rapidement qu'avec les autres fonctions. La figure 3.11 montre les différences pour deux fonctions de mise à jour des poids différentes : tangente hyperbolique et *sigmoïde*. Nous voyons que le meilleur taux d'apprentissage n'est pas le même selon la fonction.

Pour conclure sur les MLP, nos expériences ont permis de sélectionner un réseau de 3 couches cachées, 100 neurones par couche, avec la fonction d'activation *softplus*, et des MFCC en entrée du réseau.

3.2.3 Paramètres propres à un réseau de neurones convolutionnel

Les réseaux de neurones convolutionnels (Convolutional Neural Network, CNN) étant constitués de couches de convolution suivies de couches denses, les paramètres que nous devons choisir sont plus nombreux que pour le MLP. Les couches de convolution sont adaptées pour analyser des entrées corrélées. Par exemple, c'est le cas des spectrogrammes donnés en entrée sous la forme d'images. Les CNN sont en effet très efficaces en reconnaissance de formes : plus de 99% de reconnaissance correcte sur des chiffres manuscrits (MNIST) LeCun98, par exemple. Le MLP peut parvenir à des résultats similaires, mais avec davantage de couches. Par exemple, les expériences de reconnaissance de la parole grand vocabulaire de Golik Golik15 montrent qu'un MLP de 12 couches totalement connectées atteint les mêmes performances qu'un CNN construit avec une seule couche de convolution et cinq totalement connectées.

Pour la construction d'un CNN, les paramètres à ajuster sont les suivants :

— **Les paramètres d'entrée**
— **Les caractéristiques des couches de convolution** : le nombre de couches, le nombre de filtres pour chaque couche, leur taille et les fonctions d'activation.
— **Les caractéristiques des couches denses** : le nombre de couches, le nombre de neurones pour chaque couche et les fonctions d'activation.

Influence des paramètres d'entrée

Les paramètres acoustiques pouvant être utilisés en entrée d'une couche de convolution sont nombreux. Parmi ces paramètres, nous avons envisagés les MFCC, les bancs de filtres et le signal brut. Des travaux ont montré qu'utiliser le signal brut en entrée donnait des résultats légèrement moins bons pour un réseau plus complexe palaz13.

Les bancs de filtres sont corrélés en temps et en fréquence et les MFCC seulement en temps. Ce sont les bancs de filtres que nous privilégions par la suite pour les réseaux convolutionnels.

Nous avons testé différents tailles de contexte pour un réseau de neurones convolutionnel composé d'une couche de convolution suivie de deux couches denses. Les résultats sont donnés dans la figure 3.12 et nous ont permis de choisir un contexte de 25 fenêtres de voisinage, i.e. 12 fenêtres à gauche et à droite de la trame analysée.

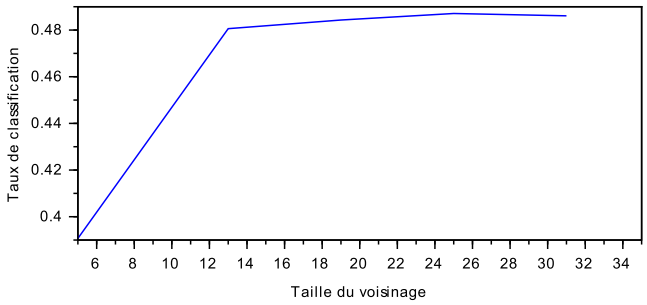

FIGURE 3.12 – Taux de classification obtenu selon la taille du voisinage pour un CNN

Influence des paramètres des couches de convolution

Les filtres sont des neurones qui ne regardent qu'une partie de la sortie de la couche précédente à la fois. Ils peuvent être de petits rectangles (2x2, 2x3), être plus grands, ou même être de la taille d'une des dimensions (une ligne ou une colonne) du spectrogramme d'entrée. Utiliser des filtres dont la dimension est une des dimensions de l'entrée revient à analyser l'évolution temporelle des paramètres (filtres de la taille du voisinage considéré) ou à représenter l'état de l'ensemble des paramètres à un instant donné (filtres de la taille du nombre de paramètres).

Le nombre de filtres à utiliser dépend de leur taille. Pour des filtres de petite taille (3x3 par exemple), nous avons remarqué qu'une quinzaine de filtres suffit généralement. Plus les filtres sont grands, plus il y a de filtres possibles et plus le nombre de filtres optimal est grand. En poussant à l'extrême le raisonnement, une couche de convolution dont les filtres seraient de la taille de l'entrée revient à utiliser une couche dense. Dans ce cas, le nombre de filtres idéal est le même que le nombre de neurones idéal pour une couche dense.

TABLE 3.1 – Résultats selon les filtres utilisés, avec *nt* désignant le nombre de trames voisines utilisées et *np* le nombre de paramètres (ici 40 bancs de filtres)

Filtres	(*nt*,1)	(1,*np*)	(3,3)	(3,5)	(5,3)	(5,5)	(7,7)
Résultat	48,4	47	48,9	48,5	49	49,8	50,4

Il y a un grand nombre de tailles de filtres possibles. Nous avons réalisé des expériences pour quelques-uns : des filtres de la taille du voisinage (*nt*,1) ou des paramètres d'entrées (1,*np*), de même que des filtres carrés ou rectangulaires de petite taille ((3,3), (3,5), (5,3), (5,5) et (7,7)). Pour comparer les résultats obtenus, nous

avons dû d'abord optimiser le nombre de filtres pour chaque dimension des filtres. Les meilleurs résultats ont été obtenus en utilisant un nombre de filtres se situant environ entre 1 et 3 fois son nombre de poids. Les taux de classification donnés dans la table 3.1 reportent les résultats des exemples précédents, utilisant une couche de convolution. Les meilleurs résultats ont été obtenu pour une couche de filtres de taille 7x7 et les plus mauvais par les filtres de la taille du voisinage.

TABLE 3.2 – Résultats selon les filtres utilisés pour des réseaux de deux couches de convolution (avec *nt* désignant le nombre de trames voisines utilisées)

Filtres	(*nt*,1),(7,1)	(5,3),(3,5)	(5,5),(5,5)	(7,7),(7,7)
Résultat	44,3	49,9	49,9	50,9

La table 3.2 montre que pour un même réseau (autres paramètres identiques : couches denses, TA, ...), ajouter une couche de convolution permet de gagner au mieux 1% Nous avons choisi d'utiliser cette dernière configuration par la suite.

TABLE 3.3 – Résultats selon le nombre filtres de la première et de la deuxième couche de convolution

	15	35	60	90
5	0,4583	**0,472**		
10	0,4686	**0,4738**	0,4676	0,4666
25		0,4816	**0,4842**	0,452
40			**0,4739**	0,4558

Pour les meilleurs filtres, nous avons regardé l'influence de leur nombre et reporté les résultats dans la table 3.3. Nous obtenons les meilleurs résultats avec 25 filtres pour la première couche et 60 pour la deuxième. Nous remarquons deux détails : augmenter le nombre de filtres n'améliore plus les résultats au-delà d'un certain seuil et il existe une corrélation entre le nombre de filtres de la première couche et celui idéal de la deuxième. Pour un nombre proche du nombre optimal de filtres pour la première couche (entre 10 et 25 dans notre exemple), il semble qu'utiliser plus de filtres pour la deuxième couche (deux à trois fois dans notre cas) est optimal. Cela peut s'expliquer par un plus grand nombre "d'images" issues de la première couche et que la deuxième doit traiter.

Cependant, trop augmenter le nombre de filtres crée des filtres inutiles dont les poids sont de faible valeur et ne s'activent jamais, comme le montre les graphiques de la figure 3.13. Le premier graphique, un histogramme des activations maximales des neurones pour l'ensemble des données testées, montre que lorsqu'il y a trop de filtres, beaucoup sont peu utilisés. Sur BUCKEYE-DEV, nous voyons que 7 des 30 filtres ont une activation maximale quasiment nulle.

Pour vérifier que ces filtres sont effectivement inutiles, nous avons regardé les probabilités obtenues en sortie du réseau en fonction des filtres utilisés. Lorsque le réseau utilise uniquement les 7 filtres les plus faibles, les probabilités maximales obtenues sont de 0,05 (au lieu de 1) et, après mise à l'échelle, nous voyons sur le deuxième graphique que les probabilités sorties sont non-informatives. Ce résultat justifie l'existence de techniques d'élagage des réseaux de neurones pour supprimer les

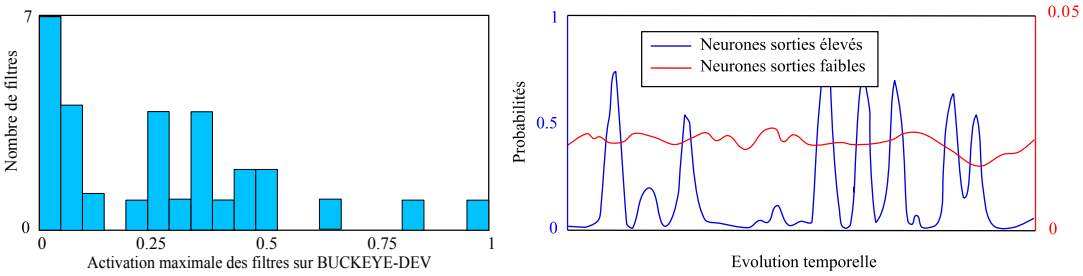

FIGURE 3.13 – Répartition des filtres selon leurs activations (*à gauche*) et résultats pour les filtres les plus faibles (*à droite*)

connexions et neurones inutiles une fois l'apprentissage terminé lecun1990optimal.

TABLE 3.4 – Influence de la fonction d'activations des couches de convolution

Filtres	*rectify*	*tanh*	*sigmoid*	*leaky rectify*	*ELU*	*softplus*
Résultat	56,8	59	59,5	56,9	58,1	57,8

Nous avons regardé l'influence de la fonction d'activation des couches de convolution et reportés les résultats dans la table 3.4. Nous voyons que la tangente hyperbolique et la *sigmoïde* obtiennent les meilleurs résultats (59%). Par la suite, nous avons choisi d'utiliser la tangente hyperbolique comme fonction d'activation des couches de convolution.

Influence des paramètres des couches denses

Pour définir la structure des couches denses qui suivent les couches de convolution, nous pouvons nous appuyer sur les expériences précédemment menées avec le MLP.

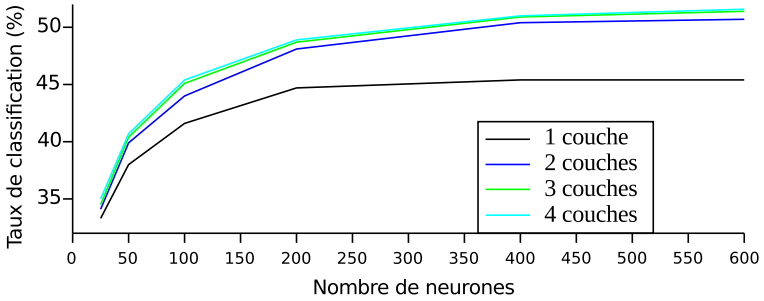

FIGURE 3.14 – Influence de la taille et du nombre de couches denses venant après une couche de convolution

La figure 3.14 montre les résultats selon le nombre de couches (de 1 à 4 couches) et le nombre de neurones par couches (de 25 à 400). Nous voyons qu'une couche est insuffisante mais qu'utiliser plus de deux couches n'est pas nécessaire sur le corpus BUCKEYE-DEV. Se limiter à 400 neurones donne les meilleurs résultats dans notre cas.

TABLE 3.5 – Taux de classification selon les fonctions d'activation des couches denses utilisées

Activation	*softplus*	*tanh*	*sigmoid*	*leaky rectify*	*ELU*
Résultat	58,5	56,5	54,4	57,4	58,2

Bien que les résultats soient similaires, nous voyons dans la table 3.5 que, comme pour les couches du réseau dense, le meilleur taux de classification est obtenu avec la fonction d'activation *softplus*.

Pour conclure sur les CNN, nos expériences ont permis de sélectionner un réseau composé de deux couches de convolutions avec des filtres 5x5 suivis de deux couches cachées de 400 neurones et d'une couche de sortie. La fonction d'activation est tangente hyperbolique pour les couches de convolution et *softplus* pour les couches denses. Les paramètres acoustiques sont des bancs de filtres avec un voisinage gauche et droite de 12 fenêtres (taille totale : 25 fenêtres).

3.3 Évaluation sur le corpus BUCKEYE-TEST

3.3.1 Réseau utilisé

Nous avons donc utilisé un réseau de neurones composé de deux couches de convolution avec des filtres 5x5 suivies de 2 couches denses de 400 neurones. La fonction d'activation des couches de convolution est la tangente hyperbolique et celle des couches denses est *softplus*, sauf la couche de sortie qui utilise softmax. La taille des sous-ensembles utilisée est de 512 exemples, le taux d'apprentissage de 0,007, la règle de mise à jour des poids est *Nesterov momentum*.

En résultat, nous obtenons environ 60% de taux de classification sur le corpus BUCKEYE-TEST : 56,8% de taux de classification au niveau de la fenêtre d'analyse (*taux_fen*) et 60,9% de taux de classification au niveau du segment phonétique (*taux_seg*). Comme vu à la section précédente, *taux_fen* sur BUCKEYE-DEV est de 58,2%. Sur le sous-ensemble d'apprentissage, nous avons obtenu un *taux_fen* de 65,1%. Les scores *taux_seg* sont environ 2% plus élevés.

3.3.2 Variabilité des résultats

Variabilité du taux de classification suivant les locuteurs

Selon le locuteur, le plus faible taux de classification au niveau des segments phonétiques est de 46% et le plus élevé est de 71%. L'écart-type des résultats entre les différents locuteurs est d'environ 5%. Nous rappelons que les locuteurs sont présents à durées égales dans le corpus. Utiliser (*taux_seg*) au lieu de (*taux_fen*) permet de réduire l'impact des erreurs aux frontières entre phonèmes. Nous avons constaté qu'il y avait entre 1 et 8% de différence entre ces deux mesures selon les locuteurs, toujours au bénéfice de *taux_seg*. La prise en compte d'un contexte plus important (le segment par vote majoritaire) permet donc d'améliorer les résultats.

Variabilité du taux de classification suivant les phonèmes

Les écarts entre les taux de classification des différents phonèmes sont plus importants, comme nous le voyons sur la figure 3.15. Le taux de classification de chaque phonème est notamment corrélé à sa fréquence dans le corpus. Moins un phonème est présent, moins ses probabilités sont élevées et moins souvent il est prédit par le réseau. Les phonèmes les moins présents sont donc souvent ceux qui ont un moins bon taux de classification. Les taux de classification les plus faibles sont de 0% pour des phonèmes présents seulement en quelques exemplaires (au plus 325 segments phonétiques dans le sous-ensemble d'apprentissage).

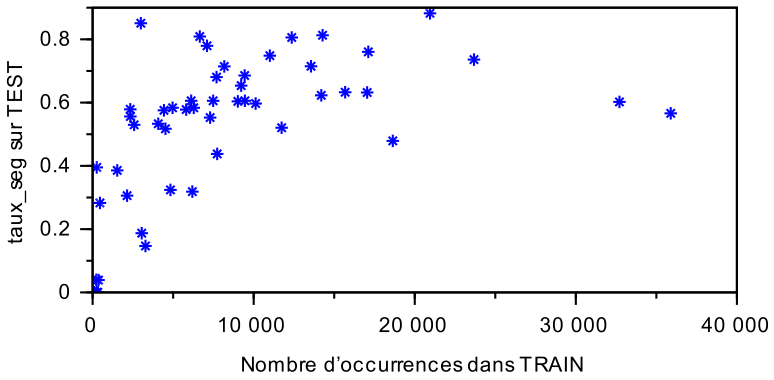

FIGURE 3.15 – Taux de classification des phonèmes en fonction du nombre d'occurrences de ceux-ci en apprentissage

3.3.3 Analyse des erreurs

Pour comprendre les points faibles de notre système, nous avons étudié quelques exemples de mauvaise attribution de classe phonétique puis regardé les erreurs de manière plus générale avec la matrice de confusion.

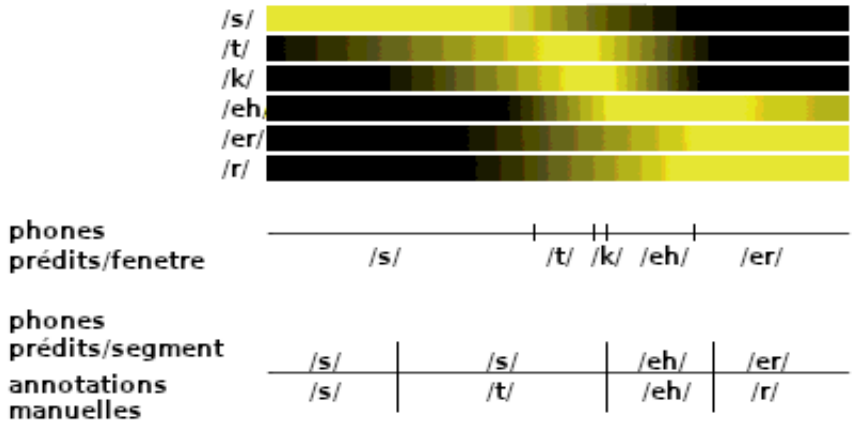

FIGURE 3.16 – Illustration de probabilités par trame (posteriorgrammes) obtenues sur un segment et analyse des erreurs par fenêtre et par segment

La figure 3.16 montre sur un exemple la vérité terrain, les prédictions par fenêtre et les prédictions par segment phonétique obtenus par vote majoritaire. Dans cet

exemple, les phonèmes prédits ont des durées différentes de la vérité terrain. Par exemple, la durée du t prédit est courte alors qu'elle est deux à trois fois plus grande selon les annotateurs humains. L'autre erreur est que deux prononciations proches sont confondues : r et ɹ.

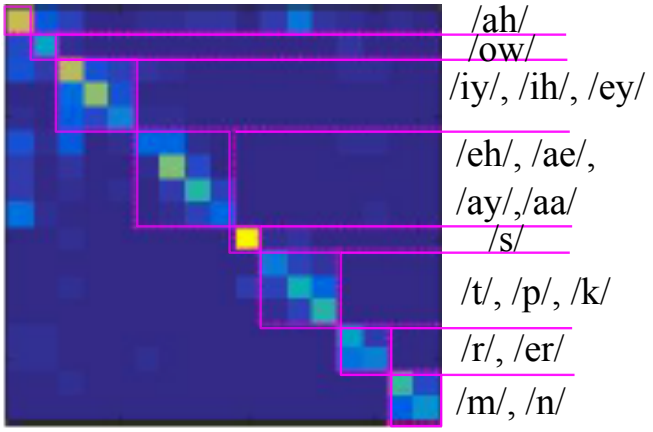

FIGURE 3.17 – Matrice de confusion des prédictions du réseau de neurones pour 17 phonèmes différents. Après regroupement des phonèmes affichant les erreurs d'attribution les plus élevées, 77% des exemples sont dans le bon groupe

Pour analyser les erreurs de manière plus générale, nous avons affiché la matrice de confusion des 17 phonèmes les plus présents dans le corpus BUCKEYE. En regroupant les phonèmes selon les erreurs de classification les plus fréquentes, nous obtenons des regroupements entre phonèmes similaires, c'est-à-dire des phonèmes qui ont des traits phonétiques en commun. Par exemple, les plosives (p, t, k), les nasales (m, n), etc. sont regroupées. Le phonème faisant exception à ce résultat est le ə qui est confondu avec un grand nombre d'autres phonèmes. Ceci peut s'expliquer par le fait qu'il a une durée importante en moyenne et aussi par le fait qu'il est le phonème le plus fréquent dans le sous-ensemble d'apprentissage en nombre de fenêtres.

La matrice de confusion montre que des phonèmes de classes phonétiques éloignées par rapport au phone ə par exemple (première ligne), comme les plosives (p principalement), mais fréquemment voisines temporellement (formant des syllabes), sont souvent confondues. De manière symétrique, la première colonne montre que cette voyelle est souvent confondue avec d'autres voyelles.

La matrice de confusion montre donc des erreurs similaires à celles de l'exemple donné figure 3.16 : des confusions entre phonèmes proches phonétiquement ou temporellement.

3.3.4 Étude de la robustesse et de la portabilité

Nous allons maintenant étudier la robustesse et la portabilité d'un réseau de neurones. Pour cela, nous allons utiliser le réseau précédemment construit (2 couches de convolution suivies de couches denses prenant les sorties des couches précédentes en entrée) que nous allons entraîner avec seulement un sous-ensemble de BUCKEYE-TRAIN, de même qu'un petit corpus de test en langue xitsonga (corpus NCHLT).

Étude de la robustesse

La robustesse d'un réseau à de nouvelles données est importante pour un bon fonctionnement en conditions réelles.

Nous avons donc effectué des expériences pour déterminer la robustesse de notre réseau face à peu de données d'apprentissage : par rapport aux occurrences des phonèmes et par rapport au nombre de locuteurs.

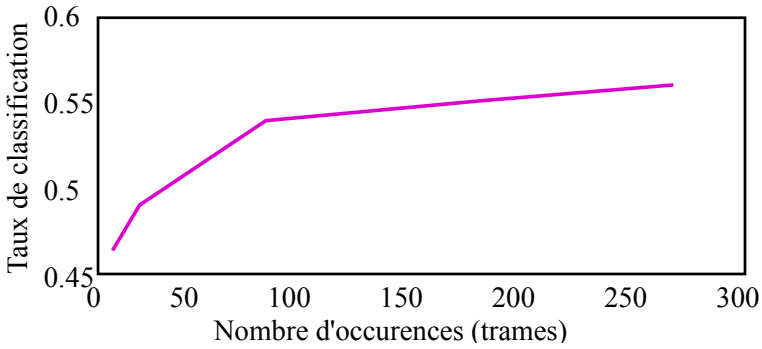

FIGURE 3.18 – Étude de la robustesse à des ensembles d'apprentissage de faible taille : taux de classification en fonction du nombre d'occurrences (trames) des phonèmes dans la base d'apprentissage

Nous avons entraîné notre réseau sur différentes tailles de base d'apprentissage. A des fins de comparaison, nous avons pris le même nombre d'occurrences pour chaque phone. Nous avons calculé le taux de classification en fonction du nombre d'occurrences (trames) des phonèmes dans la base d'apprentissage. Pour un même locuteur (voir figure 3.18), une cinquantaine d'exemples pour chaque phonème permet déjà d'obtenir un taux de classification supérieur à 50%.

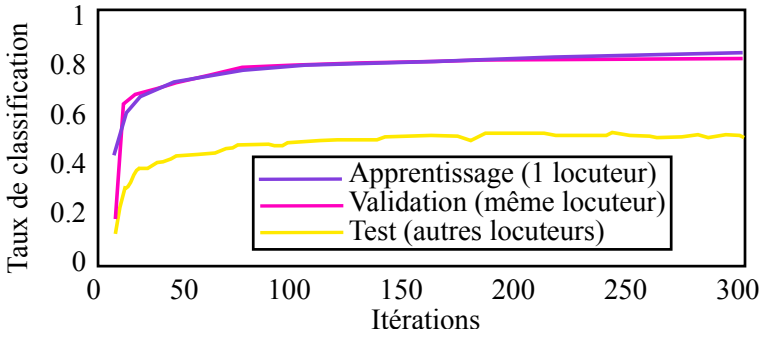

FIGURE 3.19 – Étude de la portabilité inter-locuteurs avec apprentissage sur un locuteur et test sur les autres locuteurs

Pour étudier la robustesse à de nouveaux locuteurs, nous avons pris comme exemple un cas extrême : un seul locuteur en apprentissage et tous les autres locuteurs de BUCKEYE en test. La figure 3.19 montre en fonction du nombre d'itérations d'apprentissage pour trois ensembles : l'ensemble d'apprentissage d'un seul locuteur (20 minutes), un ensemble de test du même locuteur (10 minutes) et un ensemble de test composé des autres locuteurs. Nous voyons qu'apprendre sur un seul locuteur ne permet pas une bonne généralisation à d'autres locuteurs, ce qui est attendu. De plus, dans notre expérience, la perte de performance s'élève à 20% environ par rapport à un système mono-locuteur. Nous avons ensuite regardé un cas moins extrême : avoir des locuteurs différents dans l'ensemble d'apprentissage et de test.

Notre score de taux de classification descend d'environ 60% à un peu plus de 50%, il perd presque 10%. Notre réseau ne semble donc pas très robuste pour prédire les classes phonétiques de segments de paroles prononcées par des locuteurs absents de la base d'apprentissage.

Étude de la portabilité

Une fois le réseau appris sur une langue, nous avons cherché à savoir s'il pouvait être utile pour une autre langue, par exemple une langue peu dotée comme le xitsonga (corpus NCHLT). Dans cet exemple, les deux langues utilisées sont éloignées l'une de l'autre (anglais américain et langue d'Afrique du sud).

Les classes phonétiques de ces deux langues étant différentes, les prédictions apprises avec les notations du corpus BUCKEYE ne peuvent pas être directement utilisées pour les notations du corpus NCHLT. Il va donc falloir déterminer des groupes d'occurrences puis en calculer la pureté. Pour obtenir des groupes, nous pouvons utiliser les classes prédites par la dernière couche même si elles ne correspondent pas aux annotations du corpus NCHLT, mais nous avons remarqué que les probabilité obtenues en sorties étaient faibles, avec de grandes différences d'intensité selon les classes. De plus, la dernière couche étant la couche prédisant les classes propres à l'anglais, elle est peut-être trop spécialisée sur les annotations du corpus BUCKEYE pour être utile à d'autres langues. Les activations des couches précédentes sont davantage portables à d'autres langues, mais les valeurs des activations ne donnent pas directement des groupes d'occurrences pour mesurer leur intérêt.

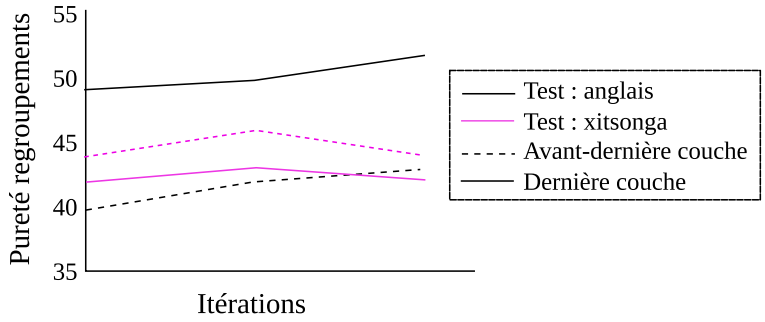

FIGURE 3.20 – Étude de la portabilité inter-langues des paramètres selon la couche dont ils sont issue (dernière ou avant-dernière) pour l'anglais (en noir) et le xitsonga (en mauve)

Pour obtenir des groupes à partir des posteriorgrammes (probabilités a posteriori, ici obtenues en sortie d'une couche de neurones), nous devons utiliser une méthode de regroupement non supervisée. Dans cette expérience, nous avons utilisé des k-means appliqués aux sorties de la dernière et de l'avant-dernière couche de notre réseau de neurones. Nous avons comparées les valeurs obtenues pour l'anglais et le xitsonga dans la figure 3.20. Comme attendu, la dernière couche permet d'obtenir de meilleurs regroupements que l'avant-dernière pour l'anglais (la langue ayant servi pour l'apprentissage) et c'est le contraire avec le xitsonga : la dernière couche est davantage spécialisée pour l'anglais.

Nous avons ensuite entraîné la dernière couche, puis ré-entraîné l'avant-dernière couche, sur le corpus xitsonga. Les résultats obtenus ont permis de compléter le graphique précédent pour obtenir la figure 3.21. Réapprendre les dernières couches du réseau avec le xitsonga a fait chuter les résultats obtenus pour l'anglais. Nous pouvons donc avancer que les dernières couches d'un réseau de neurones appris

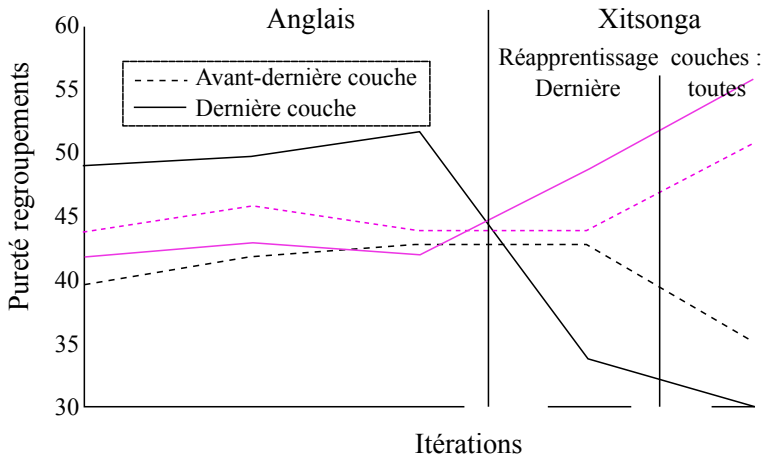

FIGURE 3.21 – Étude de la portabilité inter-langues, avec les résultats sur l'anglais en noir et ceux sur le xitsonga en mauve

sur une seule langue sont trop spécialisées pour être portables à une langue cible éloignée.

3.4 Conclusion

En conclusion, les expériences réalisées ont permis de mieux connaître les réseaux de neurones et d'obtenir un taux de classification de 60%.

Les paramètres d'entrées sont les MFCC pour les MLP et les bancs de filtres pour les CNN. Le paramètre le plus difficile à ajuster est peut-être le taux d'apprentissage, dont la valeur idéale peut changer à chaque changement du réseau. Au contraire, la taille de sous-ensembles et la fonction d'optimisation de mise à jour des poids n'ont pas eu un grand impact sur les résultats.

La structure idéale d'un réseau de neurones comporte visiblement au moins deux couches denses, que ce soit un MLP ou un CNN. Le nombre de neurones dépend du nombre de couches et d'autres paramètres, tels que l'utilisation de dropout, mais se situe généralement au-delà de la centaine.

Avoir trop de neurones ou trop de filtres amène à un problème de sur-apprentissage qui permet d'atteindre les 100% sur plusieurs heures de bases d'apprentissage (BUCKEYE-TRAIN).

L'état de l'art du taux de reconnaissance phonétique se situe vers 80% sur TIMIT (corpus de parole lue) travadi2015ensemble. Nous obtenons environ 60% sur BUCKEYE (corpus de parole conversationnelle). En comparaison, une recette Kaldi utilisée par les organisateurs du challenge Zero ressource Speech Challenge pour leur *topline* a donné un taux de classification phonétique de 73,6% versteegh2015zero.

Nous avons vu qu'apprendre sur un seul locuteur ne suffit pas pour prédire correctement les phonèmes d'autres locuteurs, mais il y a en moyenne 5% d'écart-type entre les différents locuteurs non utilisés en apprentissage, montrant une bonne robustesse au locuteur pour une base d'apprentissage suffisante. De même, apprendre sur une seule langue pour tester sur une langue éloignée ne suffit pas. Par contre, peu d'exemples par phonèmes (moins d'une centaine) permettent déjà d'obtenir des prédictions intéressantes.

En ouverture, nous avons regardé une perspective possible pour un travail futur : utiliser directement le signal brut en entrée d'un réseau de neurones.

De récentes recherches utilisent le signal brut en entrée, comme par exemple le réseau à convolutions à trous Wavenet van2016wavenet. Partir directement du signal brut est plus difficile. Lorsque le réseau apprend directement dessus, les poids des filtres de la première couche de convolution ressemblent à ceux présentés par la figure 3.22.

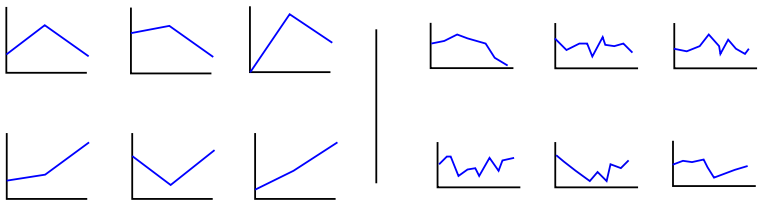

FIGURE 3.22 – Formes des filtres appris par une première couche de convolution sur du signal audio brut

Ces données sont de beaucoup plus grande taille que les bancs de filtres ou les MFCC et n'ont pas été optimisées pour le traitement de la parole. Les réseaux les utilisant en entrée sont donc plus gros, avec un temps d'apprentissage beaucoup plus long.

Utiliser le signal brut en entrée pose un autre problème : les valeurs d'entrée sont dépendantes de la phase de chaque segment de parole. Des filtres sont nécessaires pour transformer cette entrée dans une base proche de bancs de filtres palaz2015analysis. C'est donc un CNN avec une couche de convolution 1D qu'il faut nécessairement utiliser si l'entrée est le signal brut.

Partir du signal brut nécessite davantage de transformations et donc un réseau plus complexe qui ne conviendra probablement pas pour une tâche plus difficile comme l'apprentissage non supervisé. Il serait tout de même intéressant de réaliser des expériences en non-supervisé avec de tels réseaux, par exemple en s'inspirant de Wavenet et du plus récent Sincnet ravanelli2018speaker qui tente de découvrir des filtres fréquentiels pertinents pour la parole.

Chapitre 4

Segmentation en phonèmes

Sommaire

4.1 Introduction

4.1.1 Segmentation

La segmentation audio est le processus, humain (cognitif) ou automatique (quand il est réalisé par une machine), qui vise à identifier des frontières entre des unités (accords, notes, mots, syllabes, phonèmes, etc.) dans un enregistrement ou un flux sonore. En parole, c'est un sous-problème qui a diverses applications en reconnaissance automatique de la parole (RAP). Si la plupart des systèmes de RAP segmente le signal de parole en entrée en trames régulièrement espacées dans le temps (typiquement des trames de 20 ms avec un décalage de moitié), il existe des systèmes fondés sur une segmentation préalable en zones homogènes. Ces segmentations peuvent être réalisées, sans *a priori*, directement sur le signal (ANDRE-OBRECHT 1988), mais également plus ciblées sur le contenu, telles les segmentations parole/non-parole appelées détection d'activité vocale (qui a un long historique, par exemple (GERVEN et XIE 1997) et suscite un regain d'intérêt récemment comme par exemple (GELLY et GAUVAIN 2018)), segmentation en locuteurs (ROUVIER et al. 2013), segmentation en genres (homme/femme/enfant)...

Certains modèles de découverte non supervisée de sous-unités lexicales (C.-y. LEE et J. GLASS 2012 ; C.-T. CHUNG, CHAN et L.-s. LEE 2013 ; SIU et al. 2014 ; KAMPER, LIVESCU et GOLDWATER 2017), de même que des outils d'annotation de corpus (MUMTAZ et al. 2014), peuvent eux aussi utiliser une segmentation en sous-unités lexicales.

Actuellement, la recherche automatique de segments permettant d'identifier des mots ou des unités sous-lexicales est portée par l'intérêt pour l'apprentissage non-supervisé de ces unités, soit pour construire un lexique de prononciation en identifiant les mots et l'inventaire de phonèmes sans connaissance linguistique *a priori* (C.-y. LEE, O'DONNELL et J. GLASS 2015), soit pour faire des liens avec l'humain et l'acquisition du langage, en particulier par les enfants (JANSEN, DUPOUX et al. 2013). Dans notre contexte de recherche d'unités sous-lexicales, nous souhaitons évaluer la pertinence d'une segmentation préalable à cette tâche. Une évaluation spécifique sera présentée dans le chapitre 6.

4.1.2 Plan

Dans ce chapitre, nous allons aborder la segmentation automatique en phonèmes en modélisant les frontières de segments, plus précisément nous traitons cette tâche comme un problème de classification binaire présence / absence de frontière au niveau des trames acoustiques. Nous comparons les performances obtenues par des réseaux denses (MLP) et par des réseaux convolutionnels (CNN) sur le corpus BUCKEYE. Après une brève description de notre système dans la section 4.2 et des métriques d'évaluation en section 4.3, nous comparons dans la section 4.4 différentes architectures (nombre de neurones, de filtres...). Nous illustrons également l'influence des données utilisées (faible quantité, langue différente telle que le xitsonga) lors de l'apprentissage des réseaux de neurones.

4.2 Description du système

Le schéma 4.1 représente les différentes étapes de notre système permettant d'obtenir la segmentation phonétique du signal de parole. Les trois étapes sont détaillées dans les sous-sections suivantes.

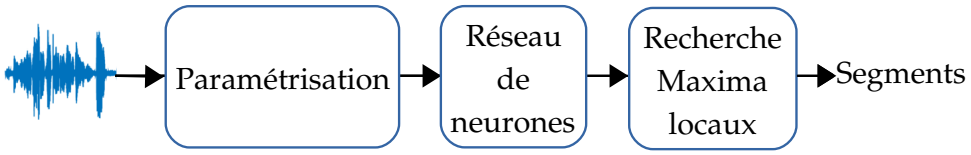

FIGURE 4.1 – Schéma du système de segmentation phonétique

4.2.1 Paramétrisation

De manière similaire au chapitre précédent, nous avons testé différents paramètres temporels et fréquentiels. Après expériences, nous avons opté pour des bancs de filtres, calculés sur le signal découpé en fenêtres de 16 ms, avec un pas de 4 ms. Utiliser un faible décalage entre les trames nous a permis d'obtenir l'emplacement des frontières avec une plus grande précision.

4.2.2 Réseau de neurones

Dans ce chapitre, nous comparons deux types de réseaux de neurones : CNN et MLP. Leur objectif est de reconnaître les variations dans les bancs de filtres marquant un changement de phone.

Le réseau de neurones réalise la tâche de segmentation comme une tâche de classification binaire : présence / absence de frontière. En général, pour attribuer une classe à une occurrence, le modèle calcule la probabilité de chaque classe puis donne en sortie la classe la plus probable. Cependant, cette dernière étape rencontre deux difficultés. D'une part, les deux classes (présence, absence de frontière) étant réparties en des proportions inégales (environ 1/20 de frontières avec un pas de 4 ms dans le corpus BUCKEYE), les probabilités en sortie sont moins favorables à la présence d'une frontière. D'autre part, lorsqu'une trame a une probabilité élevée d'être une frontière, alors ses voisines ont de grandes chances d'avoir elles aussi cette probabilité élevée.

Plutôt que d'utiliser telles quelles les prédictions binaires brutes, nous traitons les probabilités en sortie du réseau de neurones. Afin d'identifier les bornes des segments phonétiques, nous utilisons une méthode de recherche de maxima locaux.

4.2.3 Recherche de maxima locaux

La figure 4.2 illustre le processus de recherche des maxima locaux. Pour chaque fenêtre d'analyse, le réseau de neurones calcule une probabilité que celle-ci contienne une frontière (passage d'un phone à un autre). Chaque enregistrement donne lieu à une courbe de probabilités (200 valeurs sur la figure 4.2). Pour éviter de détecter des variations locales dues au bruit, nous lissons la courbe à l'aide d'une convolution par une fenêtre de Hamming de petite taille (5 échantillons, dans notre cas). Nous détectons ensuite les sommets (maxima locaux) et nous ne conservons que ceux supérieurs à un seuil. La valeur du seuil peut varier selon les besoins de privilégier la précision, le rappel ou la F-mesure (cf. section 4.3). Suite à nos expériences, comme nous le verrons dans la section 4.4.4, le seuil maximisant la F-mesure correspond approximativement à 12 phonèmes par seconde pour le corpus de parole conversationnelle BUCKEYE et 9 phonèmes pour le corpus lu NCHLT.

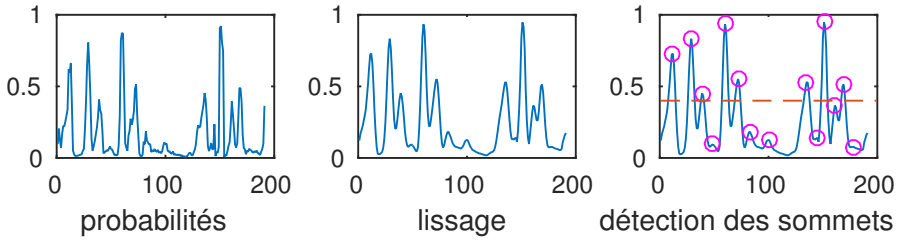

FIGURE 4.2 – Illustration de notre recherche de maxima locaux sur un enregistrement constitué de 200 fenêtres d'analyse

4.3 Métriques d'évaluation

Comme toute tâche de segmentation, une certaine marge d'erreur est tolérée lors de l'attribution de la frontière. Nous avons appliqué deux marges différentes : la marge la plus courante dans la littérature (20 ms) et une marge plus petite (10 ms). Pour évaluer les résultats, nous avons utilisé les métriques classiques de précision, rappel et F-mesure. Selon le seuil choisi pour la recherche des maxima locaux, nous repérons plus ou moins de frontières, privilégiant la précision ou le rappel. La F-mesure est une mesure qui combine le rappel et la précision. Soit P l'ensemble des éléments prédits comme frontières et F l'ensemble des éléments correspondant à des frontières :

$$\text{Rappel} = \frac{|P \cap F|}{|F|} \tag{4.1}$$

$$\text{Précision} = \frac{|P \cap F|}{|P|} \tag{4.2}$$

$$\text{F-mesure} = 2 * \frac{\text{Rappel} * \text{Précision}}{\text{Rappel} + \text{Précision}} \tag{4.3}$$

Les courbes DET (*Detection Error Trade-off*) ont en abscisse le taux de faux positifs et en ordonnée le taux de faux négatifs. Ces courbes permettent de visualiser facilement les différents résultats selon les seuils testés (MARTIN et al. 1997). Un exemple de tracé optimal, dont nous souhaitons nous rapprocher durant les expériences et donc qui serait obtenu pour une classification parfaite, serait une courbe en angle droit épousant l'axe des abscisses et des ordonnées.

4.4 Expériences

Dans cette section, nous allons tout d'abord étudier les difficultés propres à la segmentation (variabilité des durées de phonèmes, faible proportion de trames représentant une frontière), puis nous réalisons des expériences pour optimiser l'architecture de notre modèle avant d'analyser les résultats et d'appliquer notre modèle à une langue peu dotée.

Les expériences sont menées sur le corpus BUCKEYE, divisé en 3 sous-ensembles constitués d'enregistrements issus de locuteurs différents : BUCKEYE-TRAIN (1/2 du corpus), BUCKEYE-DEV (1/4 du corpus) et BUCKEYE-TEST (1/4 du corpus).

4.4.1 Analyse du problème

La durée d'une réalisation d'un phone est variable. Par exemple, parmi les réalisations contenues dans BUCKEYE-TRAIN, les plus petits segments ont une durée inférieure à 5 ms et les plus grands dépassent les 200 ms. La durée médiane est de 80 ms. La figure 4.3 montre la répartition des durées des réalisations des phonèmes pour le corpus BUCKEYE-TRAIN. Nous voyons que la majorité des segments ont une durée comprise entre 20 et 150 ms.

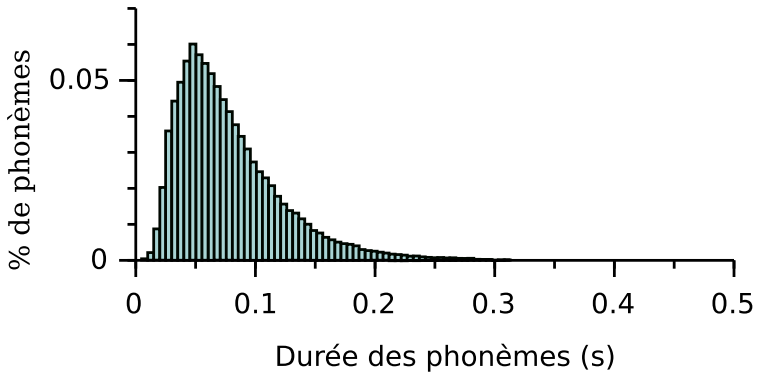

FIGURE 4.3 – Histogramme des durées des différentes réalisations de phonèmes

Cette grande variation des durées pose plusieurs problèmes. Tout d'abord, certains phonèmes sont trop courts, car proches de notre pas d'analyse acoustique de 4 ms. De plus, comme 95% des fenêtres ne correspondent pas à une frontière entre deux phonèmes, nous avons donc une importante différence de nombre d'éléments entre nos deux classes (frontière *vs.* non frontière). Ceci va biaiser l'apprentissage et encourager le réseau à privilégier la classe non frontière. Des solutions existent pour remédier à ce problème. Par exemple, nous pouvons dupliquer l'ensemble des exemples frontière pour que les deux classes aient le même nombre d'éléments. Dans notre cas, nous avons étendu les étiquettes frontière à quatre trames consécutives (deux précédentes, deux suivantes), qui contiennent elles aussi les frontières à cause du recouvrement 3/4.

4.4.2 Comparaison de différentes architectures sur BUCKEYE-DEV

Pour comparer les CNN et les MLP, nous avons optimisé sur BUCKEYE-DEV les paramètres pour chacun de ces deux réseaux (nombre de couches, de neurones, dimension des filtres de convolution, etc.) et ainsi faire les choix les plus pertinents.

Structure des réseaux

Suite au choix d'ajouter la dérivée des bancs de filtres en entrée du MLP, celui-ci s'est montré peu sensible à son nombre de couches et nous avons restreint ce nombre à 3 couches cachées. Cependant, le nombre de neurones a eu beaucoup plus d'influence, avec un maximum de performance autour de 300 neurones (cf. figure 4.4), mais pour une augmentation de seulement 1% de la F-mesure, par rapport à un modèle uniquement constitué de 50 neurones. Le CNN est optimal, quant à lui, entre 50 et 400 neurones pour les couches totalement connectées. Le nombre de filtres de ses couches de convolution a un impact de l'ordre de 1% à 2%, en absolu. Passer de 15 à 120 filtres permet ainsi de gagner 1,2%.

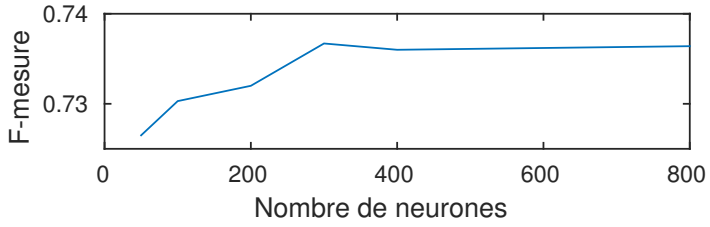

FIGURE 4.4 – Évolution de la F-mesure en fonction du nombre de neurones des couches cachées du MLP sur BUCKEYE-DEV

Prise en compte du contexte en entrée des réseaux

Le nombre de fenêtres voisines considérées s'est avéré être l'un des paramètres les plus importants : les changements de phonèmes se repèrent notamment grâce au contexte. Le MLP supporte moins bien l'augmentation du nombre de données (allant avec l'augmentation du nombre de voisins) que le CNN, dont les couches de convolution effectuent visiblement un premier traitement pertinent et réducteur. La figure 4.5 illustre l'importance de la taille du contexte pour la tâche de segmentation : les résultats s'améliorent de manière notable avec l'augmentation du nombre de voisins. Nous avons choisi 18 voisins (84 ms), l'augmentation au-delà étant très faible par rapport à l'augmentation de la complexité.

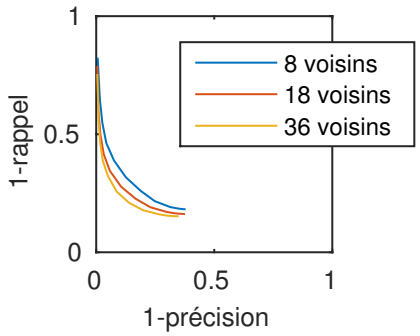

FIGURE 4.5 – Courbe DET avec un CNN en fonction du voisinage considéré sur BUCKEYE-DEV

La taille du voisinage étant un paramètre influent sur le réseau, nous comparons nos deux modèles (CNN et MLP) en fonction de son évolution. Sur la figure 4.6, nous voyons que le CNN s'avère plus efficace que le MLP : nous l'utiliserons donc uniquement celui-ci dans la suite de ce travail.

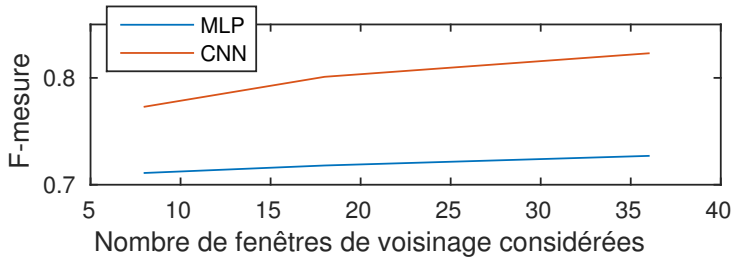

FIGURE 4.6 – Comparaison des F-mesures du MLP et du CNN selon le nombre de fenêtres de voisinage sur BUCKEYE-DEV

Les dérivées première et seconde des bancs de filtres sont souvent utilisées dans les systèmes ASR comme paramètres d'entrée. Nous avons testé leur utilité pour notre tâche mais elles n'ont fourni aucune information supplémentaire au CNN par rapport à l'utilisation de fonctions statiques uniquement. Pour comprendre pourquoi, nous avons étudié les filtres de la première couche de convolution du réseau de neurones. La figure 4.7 montre cinq exemples de filtres, de taille 2x3. Il est clairement visible que ces filtres effectuent un calcul proche d'un calcul de dérivation. Les 5 exemples donnés effectuent approximativement les calculs suivants, pour un temps t et un signal s :

1. $s(t) - s(t+1)$
2. $s(t-1) - s(t+1)$
3. $-\frac{1}{2}s(t-1) - \frac{1}{2}s(t) + s(t+1)$
4. $s(t-1) - \frac{1}{2}s(t) - \frac{1}{2}s(t+1)$
5. $s(t-1) - \frac{1}{2}s(t) + s(t+1)$.

Les différentes approximations de ces dérivées sont apprises par le modèle directement à partir des données d'entrée. Les paramètres statiques semblent donc être les paramètres d'entrée les plus appropriés pour notre tâche.

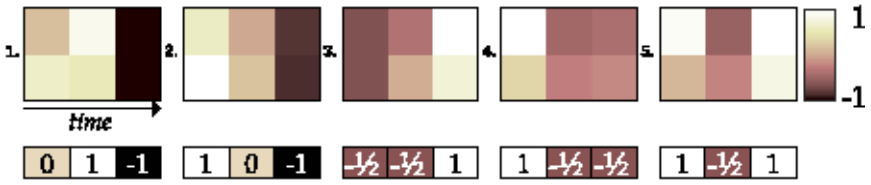

FIGURE 4.7 – 5 exemples de filtres approximant une dérivée issus de la premère couche de convolution du CNN

Réseau obtenu

En utilisant les bancs de filtres en entrée, le CNN s'est montré pertinent (taux d'apprentissage = 0,007 et momentum = 0,9). Le MLP a eu, quant à lui, besoin de la dérivée des bancs de filtres pour obtenir des résultats intéressants avec un modèle peu profond. Cette dérivée n'a pas été pertinente pour le CNN et une étude a montré que sa première couche de convolution effectuait elle-même plusieurs approximations d'une dérivée temporelle.

Suite à l'optimisation des paramètres pour chacun des deux réseaux (nombre de couches, de neurones, dimension des filtres de convolution), nous avons opté pour un CNN composé de 2 couches de convolution dotées de 60 filtres et d'une couche totalement connectée de 200 neurones.

4.4.3 Résultats sur BUCKEYE-TEST

Avec le réseau précédemment décrit, nous avons obtenu une F-mesure de 68% pour une tolérance de 10 ms. Nous pouvons obtenir une précision proche de 90%, si nous acceptons de ne trouver qu'un tiers des frontières, ou bien un rappel de 72% avec la moitié des détections erronées (cf. table 4.1).

La figure 4.8 est un exemple de résultat obtenu par le réseau de neurones, montrant la courbe (en couleur noire) des probabilités issues du CNN superposée au spectrogramme du signal. Les vraies frontières (vérité terrain) sont indiquées en

TABLE 4.1 – Résultats en pourcentage sur BUCKEYE-TEST pour 3 valeurs de seuil et 10 ms de tolérance

Taille médiane des segments phonétiques (ms)	Précision	Rappel	F-mesure
52	52	**72**	61
72	71	65	**68**
272	**94**	16	27

mauve (trait verticaux) et superposées au signal. Nous voyons que les valeurs élevées de la courbe correspondent généralement à des variations dans le spectre et sont corrélées avec les frontières.

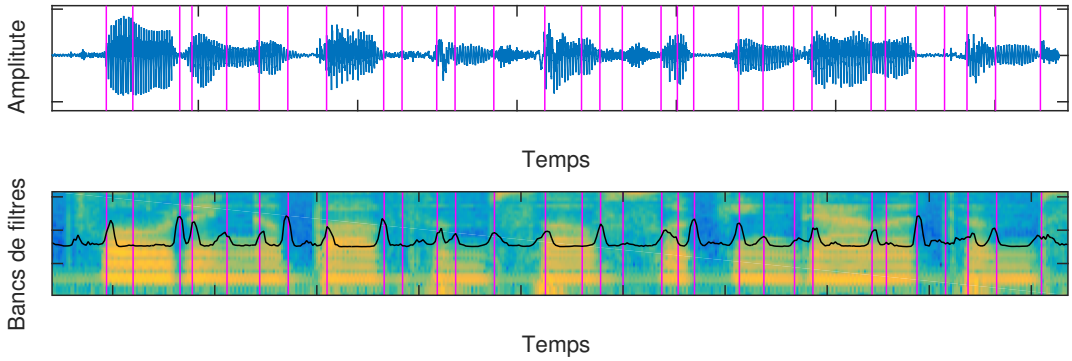

FIGURE 4.8 – Probabilités en sortie du CNN pour la segmentation sur BUCKEYE-TEST (en noir) superposé au spectrogramme (*en bas*) avec les vrais frontières phonétiques (en mauve). Le signal audio est donné dans le graphique du haut

Nous avons analysé le taux de détection des frontières de certains phonèmes parmi les plus fréquents dont certains résultats sont présentés dans la table 4.2. Les frontières des phonèmes ayant une attaque forte, comme [g] ou [k], sont plus faciles à trouver que pour [l] ou [ɹ], qui rencontrent plus de difficultés. Nous avons également observé que les frontières entre deux voyelles consécutives sont difficiles à trouver, probablement parce qu'il s'agit d'une variation lente et faible. De même, les annotateurs humains ont remarqué que la précision des frontières dépend de la taille du phone (RAYMOND et al. 2002). Par exemple, les frontières entre [oʊ] et [aɪ] sont rarement trouvées.

TABLE 4.2 – Analyse des résultats pour 5 phonèmes différents – BUCKEYE-TEST, 10 ms de tolérance

	[ɹ]	[l]	[ɔ]	[ʊ]	[g]
% débuts segments détectés	46	45	73	62	81
% fins segments détectées	49	51	39	76	81

Les résultats obtenus par notre système sont proches de l'accord inter-annotateur (calculé entre les annotateurs humains). La table 4.3 montre même que notre système

est plus précis lorsqu'il parvient à localiser une frontière : nous avons une meilleure F-mesure pour 10 ms de tolérance d'erreur et son augmentation entre 10 ms et 20 ms est inférieure à celle observée entre les annotateurs. En comparaison, pour une tolérance d'erreur de 20 ms, un modèle aléatoire obtient environ 47%.

TABLE 4.3 – Comparaison de F-mesures (%) entre l'accord inter-annotateurs et le CNN – BUCKEYE-TEST

Tolérance (ms)	Random	Annotateurs	CNN
10	26	62	68
20	47	79	79

En comparaison, un résultat de l'état de l'art atteint 78,2% en F-mesure sur un autre corpus : TIMIT, avec 20 ms de tolérance (HOANG, VU et PHI 2016).

4.4.4 Généralisation à d'autres langues

La tâche de segmentation phonétique est une tâche binaire : les données sont séparées en deux classes différentes. Nous pouvons donc espérer que peu d'échantillons peuvent suffire pour l'apprentissage de notre réseau. Nous allons aussi regarder la portabilité des modèles appris sur peu de locuteurs ou sur une langue différente de l'ensemble de test.

Il serait en effet intéressant qu'un modèle appris sur une langue avec beaucoup de données puisse aussi être utilisé pour détecter les frontières des phonèmes d'une autre langue, pour laquelle pas ou peu de données sont disponibles. Pour porter le modèle d'une langue à une autre (ici de BUCKEYE au corpus NCHLT), le seul paramètre à ajuster est le seuil appliqué à la sortie du réseau pour sélectionner les sommets. Choisir la valeur de seuil équivaut à choisir la taille médiane des segments trouvés. Comme nous pouvons le voir sur la figure 4.9, la durée moyenne optimale des segments trouvés, celle qui maximise la F-mesure, est proche de la durée moyenne véritable des données. La différence entre les deux tailles de segments optimales et réelles entre BUCKEYE et le corpus NCHLT vient notamment du fait que la parole conversationnelle (BUCKEYE) a un débit généralement plus rapide que la parole lue (NCHLT).

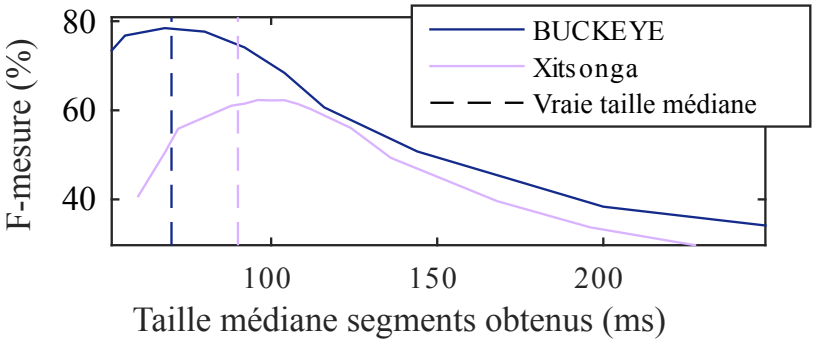

FIGURE 4.9 – Influence sur la F-mesure de la taille médiane des segments, obtenue pour différents seuils. En référence, la vraie taille médiane des phonèmes est indiquée en lignes pointillées. Ces résultats ont été obtenus avec le même réseau, uniquement appris sur BUCKEYE-TRAIN

La figure 4.10 montre la bonne adaptation du CNN à des cas peu dotés. Sur le graphique de gauche, nous remarquons qu'apprendre sur un unique locuteur donne bien évidemment des résultats légèrement inférieurs à ceux obtenus à l'aide de plusieurs locuteurs. En utilisant les données de 3 ou 5 locuteurs, nous avons obtenu des résultats très proches et l'amélioration au-delà de 5 locuteurs est presque inexistante.

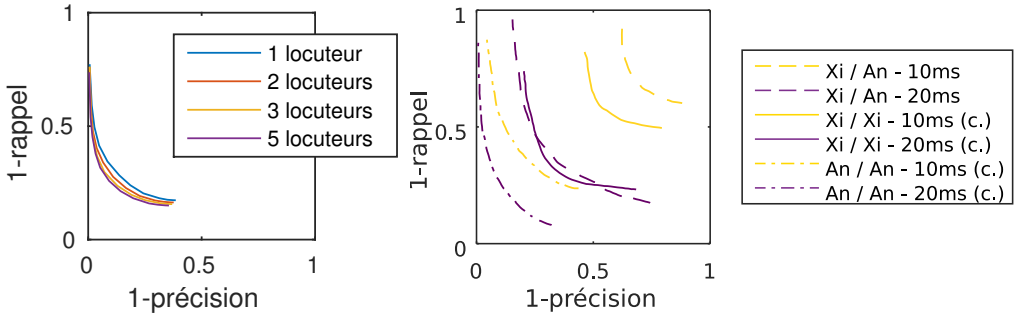

FIGURE 4.10 – Courbes DET : peu de locuteurs (*à gauche*), apprentissage sur une langue peu dotée (*à droite*). Notations : (c.) validation croisée sur peu de données selon la marge de tolérance (ms), Xi pour xitsonga, An pour anglais, [corpus de test]/[corpus d'apprentissage]

Sur le graphique de droite, se trouvent différents résultats du CNN dans le cas du xitsonga, langue peu dotée sur laquelle nous avons effectué les tests. Les résultats sur le corpus NCHLT sont moins bons que sur le corpus d'anglais dans des conditions d'apprentissage similaires (en validation croisée sur le même nombre d'échantillons, de 4 locuteurs différents). Ceci peut s'expliquer par les conditions d'enregistrement différentes : les enregistrements de xitsonga ayant été réalisés avec des smartphones, hors studio. Un résultat intéressant qui apparaît est que le modèle appris sur le petit corpus de xitsonga donne des résultats meilleurs avec une tolérance de 10 ms d'erreur que le modèle appris sur BUCKEYE-TRAIN, mais similaires pour une tolérance de 20 ms. Nous pouvons donc en conclure que le modèle appris sur le grand corpus d'anglais se montre moins précis lors de l'attribution des frontières phonétiques du xitsonga.

Afin de mieux appréhender les résultats, nous avons affiché un exemple sur la figure 4.11. Sur fond de bancs de filtres, nous avons affiché les frontières réelles (vérité terrain) des phonèmes en mauve et la courbe des probabilités obtenues en sortie du CNN en noir. Sur cet exemple, nous avons l'impression que les prédictions ont tendance à sur-segmenter.

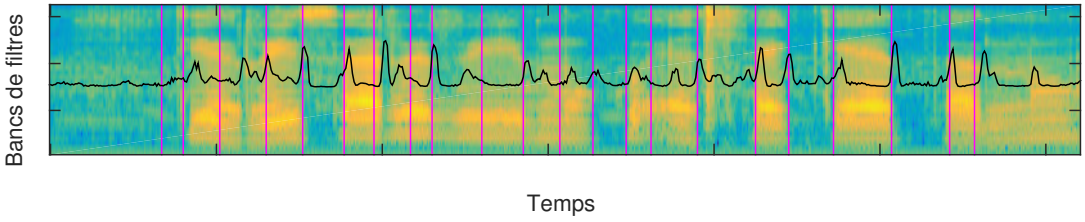

FIGURE 4.11 – Probabilités de frontières en sortie du CNN (courbe noire) appris sur BUCKEYE-TRAIN pour le corpus NCHLT, et la vérité terrain (traits mauves)

Pour améliorer la précision des résultats, nous avons adapté le modèle appris sur l'anglais avec des données de xitsonga. Pour cette expérience, nous avons découpé le corpus de xitsonga en deux sous-ensembles : Xitsonga-train pour l'apprentissage et Xitsonga-test pour le test.

La figure 4.12 représente les valeurs de F-mesure obtenues sur le corpus Xitsonga-test en fonction du nombre de minutes de données du corpus Xitsonga-train utilisées pour adapter le modèle anglais (lignes bleues) ou pour apprendre un nouveau modèle à partir de zéro (lignes mauves).

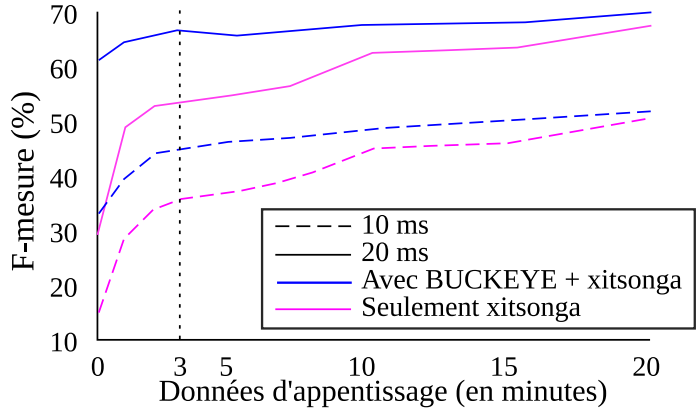

FIGURE 4.12 – Amélioration de la F-mesure selon le nombre de minutes de xitsonga utilisées pour l'apprentissage

L'ajout de quelques minutes de données d'entraînement dans la même langue que le corpus de test a grandement amélioré les résultats du modèle appris sur une autre langue, comme le montre la figure. Trois minutes seulement ont apporté une amélioration absolue de 10% de la F-mesure pour une marge de 10 ms. Pour une marge de 20 ms (lignes pleines), le modèle adapté avec au moins 3 minutes de xitsonga obtient de meilleurs résultats (65% de F-mesure) que le modèle uniquement appris sur le xitsonga (entraîné avec jusqu'à 20 minutes de données).

Il existe plusieurs façons possibles d'adapter un modèle. L'apprentissage est généralement effectué en réapprenant les dernières couches d'un réseau. Dans notre cas, le seul ré-apprentissage de la couche de sortie n'a apporté aucune amélioration. Réapprendre toutes les couches s'est avéré être la meilleure option dans notre cas, selon les tests que nous avons effectués.

Toujours en étudiant la figure 4.12, nous voyons que l'adaptation avec 20 minutes de xitsonga surpasse d'environ 2% (en absolu) le modèle appris à partir de zéro. Grâce à l'ajout du corpus Xitsonga-train, nous obtenons 52% de F-mesure pour 10 ms et 70% pour 20 ms, soit respectivement 16% et 9% de moins que BUCKEYE-TEST. Les courbes montrent que la convergence n'est pas terminée et que davantage de données pourrait améliorer les résultats. Un autre paramètre à considérer est la durée moyenne des phonèmes. En effet, la durée moyenne des phonèmes du xitsonga de NCHLT étant supérieure à celle des phonèmes anglais de BUCKEYE (90 ms contre 80 ms environ), nous pouvons supposer que cela pénalise les résultats sur le xitsonga pour une même marge d'erreur.

Nous avons continué les expériences de portabilité de notre réseau sur diverses langues en utilisant aussi le corpus de français lu BREF80 (Fr) pour les tests et l'apprentissage. La table 4.4 montre les scores obtenus pour des réseaux appris sur une ou deux langues et testés séparément sur les trois langues. L'apprentissage sur deux

langues n'a pas permis d'améliorer les résultats en test sur la troisième langue, probablement à cause de la grande différence de taille entre les trois corpus : respectivement 20 minutes pour le xitsonga (Xi), 1 heure pour le français BREF80 (Fr) et 20 heures pour l'anglais BUCKEYE (An). Nous remarquons que l'utilisation des deux langues lues durant apprentissage (Fr + Xi) permet d'obtenir de meilleurs résultats sur leurs ensembles de test.

TABLE 4.4 – F-mesure (%) obtenue avec la segmentation automatique multi-langue. Les colonnes correspondent aux corpus testés et les lignes aux corpus ayant servi pour l'apprentissage, pour 20 ms de marge d'erreur. Les résultats en gras sont ceux obtenus pour des corpus non utilisés pour l'apprentissage

	Test	An	Fr	Xi
Train	An	79	**65**	**59**
	Fr	**64**	78	**53**
	Xi	**64**	**61**	61
	An + Fr	73	74	**53**
	Fr + Xi	**64**	80	64
	An + Xi	73	**63**	58

Les différences constatées entre les trois langues sont probablement liées aux durées moyennes de leurs phonèmes. Nous avons, en effet, remarqué que les probabilités des prédictions des frontières étaient d'intensité plus faibles pour le réseau formé sur le xitsonga que sur le modèle appris sur l'anglais. Cette observation peut s'expliquer par le plus petit ratio du nombre de frontières / non-frontières dans le corpus NCHLT que dans BUCKEYE, en raison de phonèmes plus longs. Le réseau voit donc une plus faible proportion de limites et tend à attribuer des probabilités inférieures à la classe des frontières. Nous avons également constaté que les deux réseaux détectent à peu près les mêmes frontières. Malgré la différence d'échelle sur les frontières, les probabilités des deux modèles ont un taux de corrélation élevé (0,91).

4.5 Conclusion

Dans ce chapitre, nous avons décrit des résultats expérimentaux de segmentation automatique de la parole en phonèmes à l'aide de différents réseaux de neurones (CNN et MLP). Sur les enregistrements d'anglais américain issu du corpus BUCKEYE, ces modèles ont obtenu des résultats assez remarquables : 68% de F-mesure pour notre meilleur système de segmentation automatique contre 62% pour l'accord inter-annotateurs, avec une tolérance de 10 ms sur la localisation des frontières de phonèmes. De plus, les modèles ont fait preuve d'une bonne adaptation à des cas particulièrement difficiles : peu de données d'apprentissage et application à une langue différente de celle de l'apprentissage. En particulier, des performances similaires ont été obtenues sur un petit corpus de la langue peu dotée xitsonga en utilisant : 1) un modèle entraîné sur l'anglais américain, 2) un modèle entraîné sur un sous-corpus de petite taille de la langue peu dotée. Ce résultat nous fait supposer que des modèles entraînés sur des langues disposant de grandes quantités de données peuvent être utilisés avec succès pour des langues peu dotées (RENSHAW et al. 2015).

Cette segmentation automatique devrait être utile pour la découverte automatique de sous-unités linguistiques en permettant d'éliminer certaines erreurs, par exemple à l'aide d'un vote majoritaire sur les segments. Ceci sera abordé dans la section 6.4.3.

Chapitre 5

Génération de représentations par réseaux de neurones non supervisés

Sommaire

5.1 Introduction

Dans les chapitres précédents, nous avons étudié les réseaux de neurones supervisés et faiblement supervisés pour la classification et la segmentation phonétique. Nous abordons maintenant la partie non supervisée de cette thèse.

Les réseaux de neurones non supervisés les plus connus sont les auto-encodeurs, dont il existe diverses variantes. Comme vu dans le chapitre 2, un Auto-Encodeur (AE) est un réseau de neurones dont la tâche est d'apprendre à reconstruire en sortie les paramètres donnés en entrée, comme l'illustre la figure 5.1. Les deux principales utilités d'un réseau de ce type est d'obtenir de nouveaux paramètres (issus des activations d'une des couches cachées) ou de modifier les paramètres existants (en apprenant à reconstruire des paramètres différents). Pour de nombreuses variantes de l'AE, les sorties attendues ne sont en effet pas égales aux entrées. Elles peuvent par exemple correspondre à une entrée proche en temps ou en paramètres (correspondance AE, cAE (KAMPER, ELSNER et al. 2015)), à l'entrée débruitée (denoising AE, dAE (VINCENT et al. 2008)) ou encore à l'entrée traduite dans une autre langue, si le but du réseau est d'être un traducteur (SUTSKEVER, VINYALS et LE 2014).

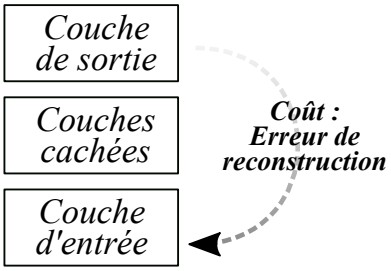

FIGURE 5.1 – Exemple de structure d'un Auto-Encodeur

Durant nos expériences, nous avons aussi tenté une approche non supervisée différente des AE. Au lieu de conserver comme paramètres la sortie du réseau ou d'une des couches cachées, nous nous sommes intéressés aux valeurs des poids du réseau, dans un esprit similaire aux paramètres LPC (Linear Prediction Coefficients). Les deux principaux atouts pouvant être attendus par l'utilisation des réseaux de neurones sont un temps réduit pour le calcul de paramètres de grande taille (grâce à la complexité en $O(n)$ pour le réseau de neurones contre $O(n^3)$ pour le calcul des LPC standards) de même que les possibilités d'amélioration offertes par les différentes structures de réseaux de neurones.

Dans ce chapitre, nous commencerons par présenter nos expériences sur les auto-encodeurs dans le but d'obtenir de nouveaux paramètres plus pertinents pour notre tâche, à partir des MFCC et des bancs de filtres puis du signal brut. Nous réaliserons ensuite des expériences sur un réseau de neurones non supervisé différent, imitant l'extraction de paramètres LPC.

5.2 Auto-Encodeurs : projection des paramètres dans un nouvel espace

Nous allons expérimenter les Auto-Encodeurs pour obtenir de nouveaux paramètres, issus d'une de leurs couches cachées. Comme les autres réseaux de neurones, les AE effectuent plusieurs projections non linéaires avec les données d'entrée pour

les transformer, jusqu'à obtenir les sorties souhaitées. Utiliser parmi les couches cachées une couche de plus petite taille (appelée couche de Bottleneck) permet en théorie de compresser l'information utile. Nous avons commencé à prendre en main les AE par des tests simples (utilisation de seulement deux classes phonétiques éloignées : une fricative et une voyelle) qui nous ont donné des résultats encourageants, comme nous allons le voir. Nous avons ensuite optimisé les hyperparamètres de notre réseau dans le but d'obtenir de nouvelles représentations paramétriques les plus efficaces possibles pour la classification phonétique. Nous étudions ensuite leur intérêt dans un cadre de découverte de sous-unités linguistiques.

5.2.1 Exploration des AE à faible dimension

Pour faciliter la visualisation des résultats, nous avons restreint la taille des paramètres générés à deux dimensions. Nous avons commencé par des réseaux simples (trois couches cachées dont une couche centrale de taille 2), avec peu de classes phonétiques et un seul locuteur (*s01* de BUCKEYE).

Visualisation des paramètres de Bottleneck

Nous avons entraîné notre réseau sur une base de données uniquement composée des occurrences de deux phonèmes très éloignés : une fricative [s] et une voyelle [ə]. La projection des entrées sur le plan de dimension 2 généré par la couche de Bottleneck est affichée dans la figure 5.2. Les deux classes se démarquent l'une de l'autre (en violet et noir dans la figure) avec une zone de recouvrement entre les deux et quelques exemples éparpillés aux alentours. Sur cet exemple, l'auto-encodeur regroupe les échantillons selon leur classe phonétique et les paramètres qu'il génère peuvent donc potentiellement avoir une utilité pour la découverte non supervisée de phonèmes.

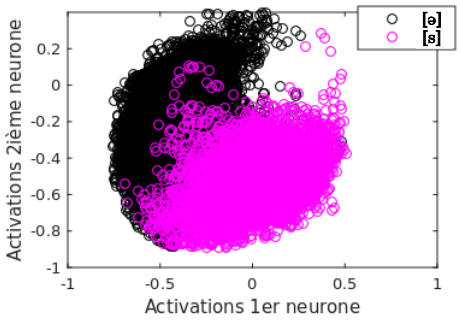

FIGURE 5.2 – Répartition des activations de la couche de Bottleneck (2 dimensions) d'un AE pour des échantillons de deux phonèmes éloignés : [ə] et [s] (notés /ah/ et /s/ dans le graphique, l'écriture d'annotation choisie dans BUCKEYE)

Nous avons ensuite utilisé l'ensemble des classes phonétiques et étudié leur répartition selon trois groupes : les consonnes fricatives, les autres consonnes, et les voyelles. Pour mieux pouvoir analyser les résultats, nous avons séparé l'affichage des trois groupes de phonèmes dans des graphiques différents. Les différentes densités de répartition de leurs occurrences sont affichées dans la figure 5.3. Ces graphiques montrent que les projections de chaque ensemble phonétique sont groupés en des endroits différents du plan, avec du recouvrement entre eux. Chaque groupe

a un sommet bien marqué, ce qui pourrait faciliter leur découverte de manière non supervisée.

Ces trois ensembles de classes sont distants les uns des autres, principalement celui de gauche (les fricatives). Néanmoins, le recouvrement des deux ensembles de droite ne permet pas une séparation propre avec seulement deux neurones.

FIGURE 5.3 – Densité des répartitions de l'ensemble des phonèmes d'un locuteur séparées en trois ensembles : les fricatives (*à gauche*), les voyelles *(au centre)* et les autres consonnes (*à droite*)

Séparation des classes phonétiques

Maintenant que nous avons vu que les paramètres de l'Auto-Encodeur permettaient de différencier certains groupes de phonèmes entre eux (voyelles, fricatives, autres consonnes), nous allons essayer de séparer ces groupes.

Nous commençons nos expériences avec deux groupes phonétiques (fricatives ou non) projetés sur un seul axe, dans le but de les séparer à l'aide d'un seuil come le font certains classifieurs. Pour ce faire, nous avons tout d'abord étudié les répartitions des valeurs des activations selon leur groupe phonétique à l'aide des histogrammes de la figure 5.4. L'histogramme de l'ensemble des éléments est affiché dans le graphique de gauche et ceux superposés des éléments des deux groupes dans le graphique de droite. Celui-ci montre qu'il n'y a pas de séparation nette entre les répartitions des deux ensembles ; une partie de leurs éléments est mélangée. La séparation ne peut donc pas être parfaite avec ce seul axe, même de manière supervisée. Néanmoins, une séparation peut être trouvée de manière non supervisée, comme nous le voyons sur le graphique de gauche : deux sommets se démarquent. La valeur optimale de la séparation entre les deux sommets, marquée d'un trait mauve sur le graphique de gauche, nous permet d'obtenir 88,7% de classification correcte.

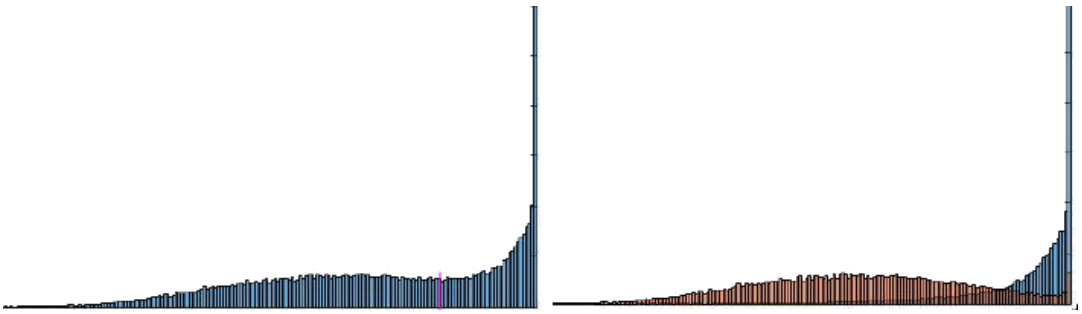

FIGURE 5.4 – Histogrammes des activations de la couche de Bottleneck d'un AE composée d'un neurone, avec l'ensemble des données à gauche et les données séparées en deux sous-ensembles à droite selon leur classe phonétique (fricatives/autres)

Mais séparer les fricatives des autres phonèmes n'est pas une tâche difficile et il n'est pas nécessaire d'utiliser un réseau de neurones pour y arriver. Pour que les

paramètres générés par notre Auto-Encodeur soient intéressants, il faut qu'ils soient utiles pour différencier un plus grand nombre de classes phonétiques.

Nous sommes donc passés à un réseau avec davantage de neurones dans la couche de Bottleneck.

Augmentation du nombre d'axes

Lorsque nous avons utilisé davantage d'axes, nous avons constaté une grande corrélation entre eux, comme illustré dans la figure 5.5. Ce résultat provenait probablement d'un problème lié à l'apprentissage et à l'ajustement des valeurs des hyperparamètres auquel nous nous sommes heurtés à de nombreuses reprises.

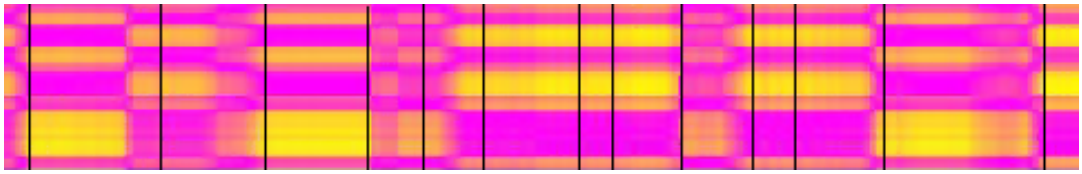

FIGURE 5.5 – Exemple de posteriorgramme représentant l'évolution temporelle des activations issues de la couche de Bottleneck de l'AE. Les frontières phonétiques issues de l'annotation manuelle sont indiquées en noir

L'information la plus intéressante apportée par ces résultats est que les valeurs des paramètres subissent des passages brusques d'une valeur extrême à l'autre. Les activations passent de 0 ou -1 à 1 en l'espace d'une ou deux trames, correspondant visiblement à certains changements phonétiques (consonne/voyelle). Ces changements brutaux de valeurs peuvent permettre d'effectuer facilement une segmentation, qui s'avère corrélée avec la segmentation phonétique manuelle, illustrée en noir sur la figure 5.5. Cette segmentation a néanmoins le défaut de sous-segmenter, dû au fait qu'elle ne sépare les segments qu'en deux groupes. Les frontières entre deux phonèmes regroupés ensembles, c'est-à-dire ayant des paramètres de Bottleneck proches, ne sont donc pas détectées.

Pour obtenir des paramètres plus intéressants, nous avons dû reprendre les expériences réalisées sur les hyperparamètres (voir chapitre 3) pour adapter la structure du réseau et les valeurs de ses hyperparamètres au cas particulier des AE.

5.2.2 Optimisation des paramètres des AE

Pour optimiser les paramètres de notre AE, nous avons besoin d'une mesure permettant de déterminer leurs meilleures valeurs. L'utilisation de méthodes de regroupement non supervisées, telle que le k-means, pour mesurer la pureté des groupes obtenus n'est pas un choix efficace : les résultats dépendent de la méthode utilisée et sont diminués à cause du facteur non supervisé des regroupements. Comme nous nous intéressons ici uniquement à la pertinence des paramètres obtenus et non à l'optimisation d'une méthode de regroupement non supervisée, nous nous sommes tournés vers une mesure supervisée. Pour mesurer l'efficacité des paramètres, nous avons donc utilisé le taux de classification phonétique, obtenu par un réseau de neurones.

Nous avons choisi un réseau dense d'une seule couche cachée qui prend en entrée une seule trame de paramètres, sans considérer de voisinage. Utiliser un réseau de petite taille a pour but de mesurer davantage les qualités des paramètres pour

la discrimination des phonèmes que les capacités d'un réseau de grande dimension. Dans cette section, nous travaillons sur le corpus BUCKEYE.

Réseau dense

Mise à jour des poids Nous avions vu dans le chapitre 3 que les différentes règles de mise à jour des poids obtenaient des résultats similaires, mais pour des taux d'apprentissage différents. Pour chaque règle de mise à jour, nous avons donc affiché l'évolution du taux de classification en fonction du taux d'apprentissage dans la figure 5.6. Elles ont toutes un maximum proche pour des valeurs différentes et pour des valeurs trop élevées de taux d'apprentissage un minimum similaire au hasard. Dans cet exemple, c'est le rmsprop qui a obtenu - de peu - les meilleurs résultats, et c'est lui que nous avons conservé par la suite.

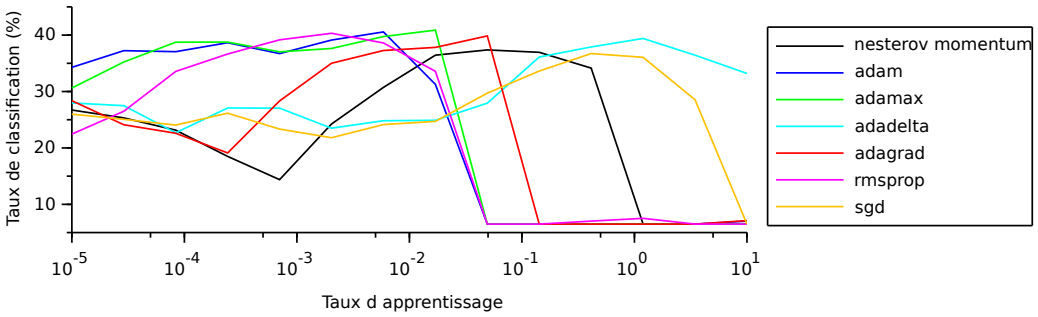

FIGURE 5.6 – Évolution du taux de classification en fonction du taux d'apprentissage et de la règle de mise à jour des poids

Fonction d'activation Dans notre contexte d'utilisation des activations des couches cachées comme nouvelle représentation paramétrique, la fonction d'activation des couches n'a pas seulement un impact sur les calculs effectués par le réseau, elle influence directement les valeurs prises par les nouveaux paramètres. L'utilisation d'un réseau de neurones supervisé sur ces paramètres permet de s'adapter aux différentes répartitions obtenues et il n'y a que 2% de différence maximale entre les différentes règles de mise à jour des poids, dont les résultats sont donnés dans la table 5.1. Les deux meilleurs résultats sont obtenus par *tanh* et *ELU*. Par la suite, nous avons utilisé *ELU*.

TABLE 5.1 – Influence de la fonction d'activation des couches denses

Fonction d'activation	*tanh*	*sigmoïde*	*rectify*	*ELU*	*softplus*
Résultat	39,7	38,7	39,5	**40,7**	39,4

Entrée du réseau Nous avions vu dans les chapitres précédents que les MLP obtenaient de meilleurs résultats avec des MFCC en entrée et c'est ce que nous avons utilisé ici. Durant nos expériences de classification supervisée, les réseaux utilisaient efficacement un grand voisinage (une vingtaine de fenêtres pour un réseau convolutionnel). La tâche d'un AE étant plus difficile que celui d'un réseau supervisé, le

réseau supporte visiblement mal un grand voisinage comme le montre les résultats affichés dans la figure 5.7. La courbe des taux de classification est maximale entre 8 et 12 fenêtres consécutives prises en entrée, nous choisissons d'utiliser un voisinage de 9 trames.

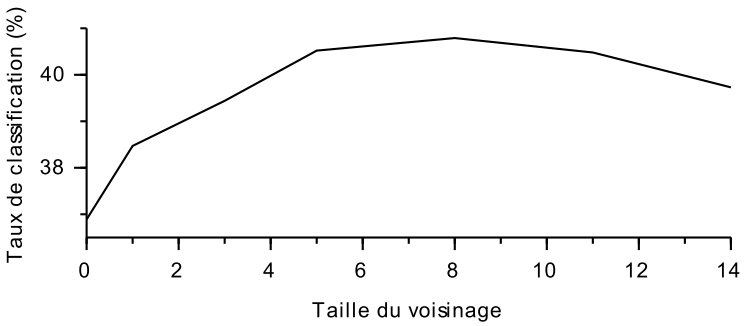

FIGURE 5.7 – Évolution du taux de classification en fonction de la taille du voisinage

Nombre de neurones et de couches Nous avons également effectués des expériences pour estimer le meilleur nombre de couches et de neurones par couches. Pour que les résultats soient comparables entre eux, il faut évaluer des paramètres de même taille, nous avons donc toujours conservé le même nombre de neurones dans la couche centrale. Les résultats obtenus sont affichés dans la figure 5.8. Nous constatons qu'augmenter le nombre de neurones n'améliore pas les résultats dans notre exemple. Par la suite, nous utilisons 5 couches denses cachées et 150 neurones par couche, hors couche de Bottleneck.

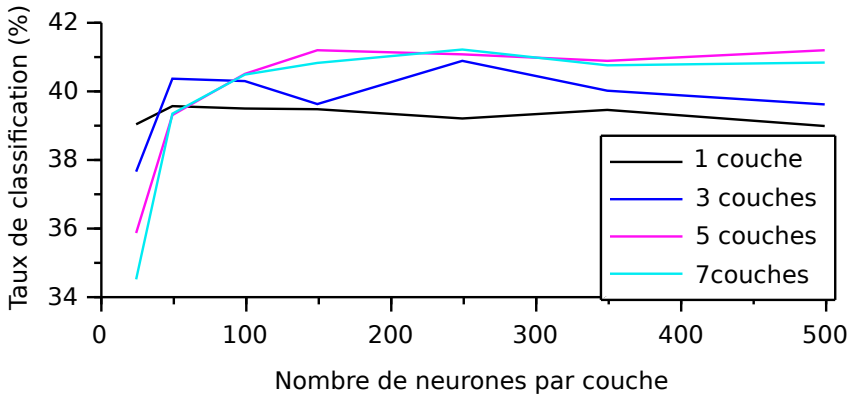

FIGURE 5.8 – Évolution du taux de classification en fonction du nombre de neurones et du nombre de couches

Réseau convolutionnel

Nous avons ajouté une couche de convolution en début de notre auto-encodeur en supposant que les filtres repéreraient plus facilement les dépendances temporelles et fréquentielles des paramètres d'entrée.

L'importance d'un bon réglage du taux d'apprentissage peut être constaté dans la figure 5.9. Par la suite, nous avons utilisé l'adam, avec un taux d'apprentissage variant selon les modifications apportées au réseau.

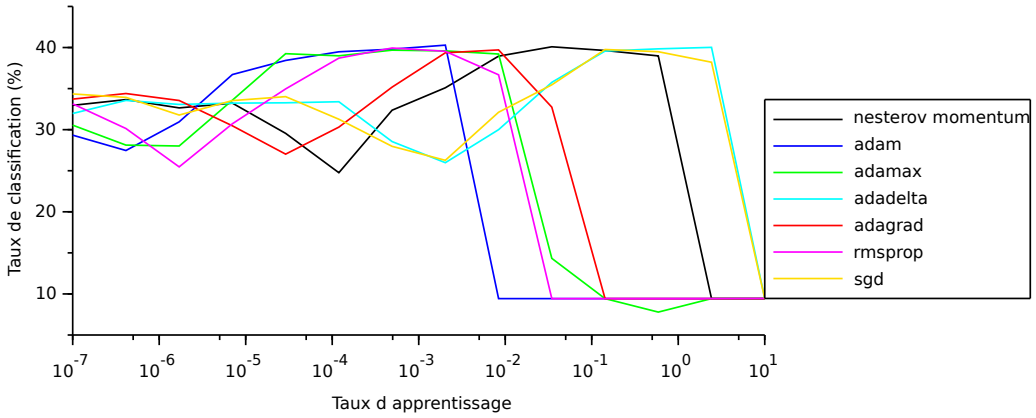

FIGURE 5.9 – Évolution du taux de classification en fonction du taux d'apprentissage et de la règle de mise à jour des poids

Contrairement aux résultats obtenus dans un cadre supervisé (chapitre 3), l'AE convolutionnel a préféré des MFCC et non pas des bancs de filtres en entrée : il a obtenu 39,8% avec les MFCC, 40,2% avec les MFCC et leurs deux premières dérivées, et seulement 31,1% avec les bancs de filtres.

Les MFCC sont des paramètres davantage modifiés que les bancs de filtres, optimisés pour le traitement de la parole. Ils ont donc besoin de moins de traitements pour des résultats similaires, ce qui permet visiblement aux AE d'être plus efficaces sans supervision pour les guider dans la tâche à effectuer.

Les paramètres générés n'obtiennent pas à chaque nouvel apprentissage le même résultat pour les mêmes hyperparamètres, à environ 0,5% près. Les écarts entre certains résultats reportés dans la table 5.2 sont donc trop faibles pour être significatifs. Nous pouvons néanmoins constater que de manière similaire au réseau précédent (uniquement constitué de couches denses), ce sont les fonctions *ELU* et *tanh* qui permettent d'obtenir les meilleurs résultats (environ 40%). Nous avons également testé avec une activation linéaire, qui a elle aussi obtenu un score un peu supérieur à 40%. Le choix de la fonction d'activation a eu moins d'importance pour la couche de convolution, comme nous le voyons dans la première ligne de la table. Par la suite, nous utilisons une tangente hyperbolique pour l'activation de la couche de convolution et la couche de Bottleneck et *ELU* pour les autres couches denses.

TABLE 5.2 – Influence de la fonction d'activations des couches denses sur le taux de classification

Fonction d'activation	*tanh*	*sigmoïde*	*rectify*	*ELU*	*softplus*	linéaire
Couches convolutionnelles	40,2	38,9	40,0	40,1	38,6	40,2
Couches denses	40,2	23,8	38,7	40,3	34,0	40,3
Couche de Bottleneck	40,5	38,8	36,2	40,1	38,6	40,5

En nous appuyant sur les résultats obtenus par les différentes tailles de filtres dans le chapitre 3, nous avons testé les quatre tailles suivantes pour la couche de convolution : (1,39), (3,3), (5,5) et (7,7). Les filtres, dont les taux de classification résultant sont reportés dans la table 5.3, ne se différencient pas entre eux et obtiennent tous un résultat similaire au réseau dense. Par la suite, nous choisissons d'utiliser des filtres de taille (5,5).

TABLE 5.3 – Taux de classification selon les tailles de filtres utilisées

Filtres	(1,39)	(3,3)	(5,5)	(7,7)
Résultat	40,1	40,0	40,0	40,3

5.2.3 Résultats

Structures et paramètres des réseaux construits

Au final, nous obtenons un AE dense et un AE convolutionnel donnant des résultats proches au regard de notre méthode d'évaluation. Les deux réseaux utilisent 9 trames consécutives de MFCC en entrée et la règle rmsprop ou adam pour la mise à jour de leurs poids.

Le réseau dense est composé de cinq couches cachées avec *ELU* comme fonction d'activation, dont une couche centrale de la taille du nombre de paramètres souhaités et les deux autres couches de 150 neurones.

Le réseau convolutionnel est composé d'une couche de convolution suivie d'une couche dense, de la couche de Bottleneck, de deux autres couches denses puis de la couche de sortie. La fonction d'activation de la couche de convolution et de la couche de Bottleneck est la tangente hyperbolique, et celle des autres couches denses est *ELU*.

Taux de classification

Les paramètres générés par notre AE permettent d'obtenir de meilleurs résultats que les MFCC avec l'utilisation de notre méthode d'évaluation, c'est-à-dire en utilisant les paramètres sans voisinage en entrée d'un petit réseau de neurones. Selon leur dimension, les paramètres générés permettent d'obtenir des scores différents. Pour une dimension égale à la dimension des MFCC d'entrée (39), ils permettent d'obtenir 41% comme nous le voyons dans la figure 5.10. En comparaison, lorsque nous utilisons directement une trame des MFCC, notre mesure obtient seulement 33,6%.

Regroupements non supervisés

Maintenant que nous avons vu l'utilité des paramètres générés par notre AE dans un cadre supervisé, nous allons étudier leur utilité dans un cadre totalement non supervisé en utilisant des méthodes de regroupement.

Une caractéristique qui sera probablement importante pour un bon fonctionnement des méthodes de regroupement non supervisées est que les échantillons des classes phonétiques soient regroupés entre eux de manière marquée. Or deux paramètres qui peuvent influencer la répartition des échantillons sont la règle de mise à jour des poids et la fonction d'activation de la couche de Bottleneck.

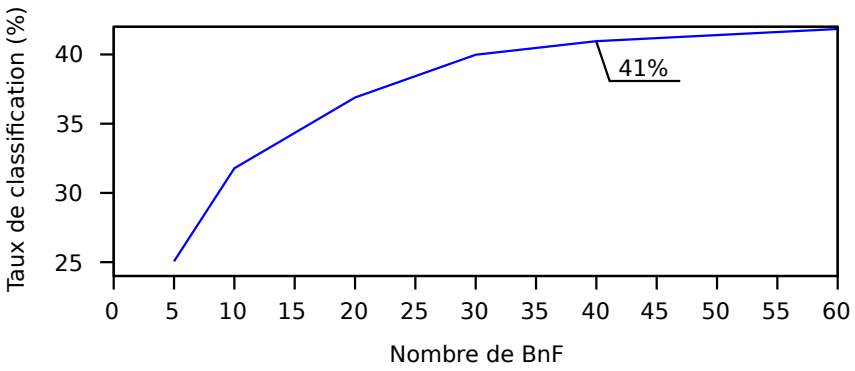

FIGURE 5.10 – Évolution du taux de classification en fonction du nombre de paramètres générés par l'AE

La figure 5.11 montre des exemples de répartition de paramètres générés (en dimension 2) selon la règle de mise à jour. Nous voyons que le momentum, l'adam, l'adamax, l'adagrad et le rmsprop obtiennent une bonne couverture de l'espace.

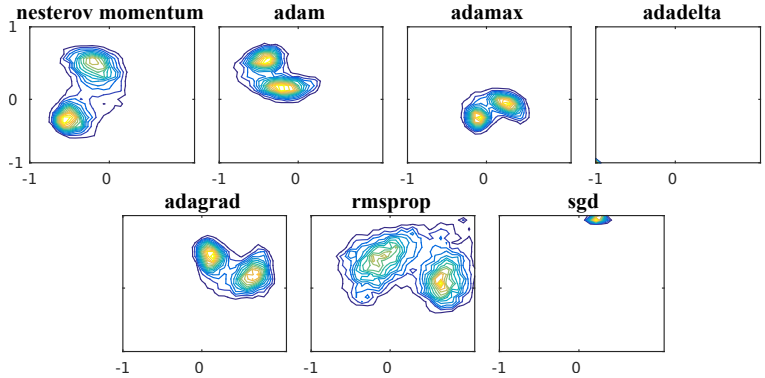

FIGURE 5.11 – Couverture de l'espace des représentations générées par l'AE selon la fonction de mise à jour des poids

Pour avoir une idée de l'utilité potentielle de nos nouveaux paramètres pour la tâche de découverte non supervisée de pseudo-phones, nous avons utilisé les k-means comme méthode de regroupement et visualisé les résultats obtenus.

Les k-means ont regroupé les échantillons en 30 groupes, dont les répartitions des classes phonétiques à l'intérieur de quelques uns des groupe est donné dans la figure 5.12. Parmi les regroupements que nous avons affichés, nous avons laissé les meilleurs groupes à gauche et ceux correspondant le moins à des groupes phonétiques à droite. Nous voyons qu'aucun groupe n'a une pureté de 100% : il y a des éléments de plusieurs classes phonétiques différentes dans chacun. Les deux groupes de droite ne laissent pas apparaître de classe phonétique dominante et contiennent peu d'éléments de beaucoup de classes différentes. Au contraire, dans les regroupements affichés à gauche, certaines classes phonétiques se démarquent nettement, comme le premier avec la classe jaune bien marquée. Ainsi, nous pouvons conclure que les paramètres permettent d'isoler en partie certaines classes.

Un indice pouvant nous permettre d'estimer l'intérêt potentiel des paramètres générés par l'AE, de même que des groupes trouvés par le k-means, est leur évolution temporelle. En effet, la corrélation entre les changements temporels d'attribution de groupes et les frontières entre les véritables phonèmes peut indiquer si les

FIGURE 5.12 – Histogrammes des répartitions des classes phonétiques pour quelques exemples de groupes obtenus par k-means sur les paramètres issus de notre AE

paramètres de l'AE regroupent des éléments linguistiques de la taille des phonèmes. Un premier élément de réponse avait été vu en début de chapitre sur la figure 5.5, les variations brusques de ces paramètres coïncidant parfois avec les frontières phonétiques.

La figure 5.13 montre, quant à elle, l'évolution temporelle des probabilités des groupes du k-means (les distances aux centroïdes). Il y a une grande variation temporelle des attributions des groupes, ce qui a un impact sur leur pureté puisque des parties d'un même phonème sont séparées dans différents regroupements. Certains groupes coïncident même avec les frontières phonétiques et non les phonèmes. Un autre problème montré par ces courbes est la proximité des probabilités de différents groupes, et l'alternance de groupes entraîne des changements d'attribution indésirables entre trames voisines.

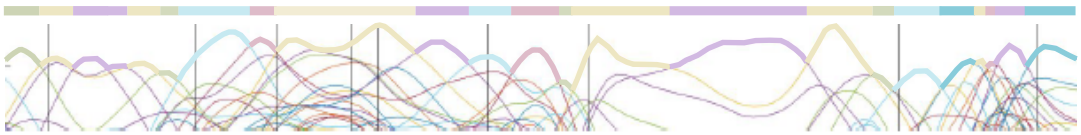

FIGURE 5.13 – Illustration de la variation temporelle des probabilités des groupes k-means. Les frontières phonétiques sont indiquées par des traits verticaux noirs et les couleurs des groupes attribués sur la ligne en haut du graphique

Jusqu'à présent, nous avons développé un AE utilisant en entrée des paramètres acoustiques optimisés pour le traitement de la parole, tels que les MFCC. Nous allons nous intéresser à un AE compressant le signal brut pour générer de nouveaux paramètres.

5.2.4 Autres expériences avec AE : la compression audio

Introduction : compression audio et réseaux de neurones

Les précédentes expériences n'ayant pas été aussi concluantes que prévu, nous nous sommes focalisés sur la génération de nouveaux paramètres directement à partir du signal brut en forçant l'AE à compresser l'information utile en réduisant sa dimension. Le résultat espéré est qu'une bonne synthèse audio signifierait de bons paramètres de Bottleneck pour le regroupement phonétique.

En choisissant de générer des paramètres de taille plus petite que ceux d'entrée, nous effectuons une compression des données. Les bancs de filtres et les MFCC sont deux exemples de paramètres audio effectuant une compression avec perte du signal audio, essayant de conserver uniquement l'information utile présente dans le signal. L'AE est un réseau approprié pour la compression de données avec perte, comme l'ont montrés différents travaux (G. E. HINTON et SALAKHUTDINOV 2006 ; THEIS et al. 2017).

Architecture du réseau

Nous avons réalisé des expériences de compression audio à partir du signal brut en utilisant des AE convolutionnels. Nous avons utilisé l'erreur quadratique moyenne (Root Mean Square, RMS) pour construire notre réseau. Des résultats d'expériences sont donnés dans la figure 5.14 : avec ou sans couche dense avant la couche de Bottleneck (à gauche) et selon le nombre de couches de convolution pour décomposer le signal (à droite). Les meilleurs résultats ont été obtenus avec 4 couches de convolution utilisant des filtres de taille 7, alternées avec une couche de maxpooling de taille 2, suivies d'une couche de Bottleneck puis les couches inverses effectuant la reconstruction du signal, soit 10 couches cachées.

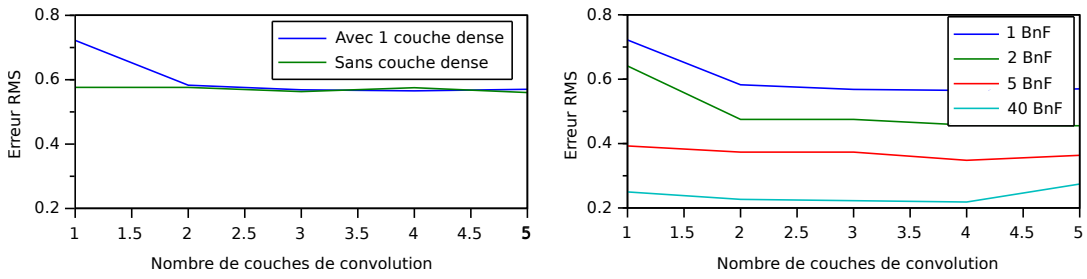

FIGURE 5.14 – Erreur RMS pour un réseau avec ou sans couche dense autour de la couche dense de Bottleneck (*à gauche*) et selon le nombre de couches de convolution (*à droite*)

Signal reconstruit

NOus avons fait écouter les signaux reconstruits à des personnes n'ayant pas écoutés les signaux originaux. La parole obtenue en synthèse était compréhensible, pour moins de 1 ko/s de paramètres, soit une compression de 85% pour un signal échantillonné à 15 000 échantillons par seconde.

Un exemple de résultat est donné dans la figure 5.15, avec le signal original (en bleu foncé) et les reconstructions pour différentes tailles de couches de Bottleneck superposées. La figure en bas à gauche est un agrandissement permettant de mieux voir les détails. L'utilisation d'un seul neurone dans la couche centrale offre la reconstruction la plus mauvaise, bien moins proche du signal original que celles avec

2 ou 5 neurones, qui donnent des résultats presque similaires sur cet exemple. Le signal reconstruit à partir de 30 neurones est évidemment la meilleure reconstruction, très proche du signal original : le taux de compression est, dans ce cas, moins élevé que les autres.

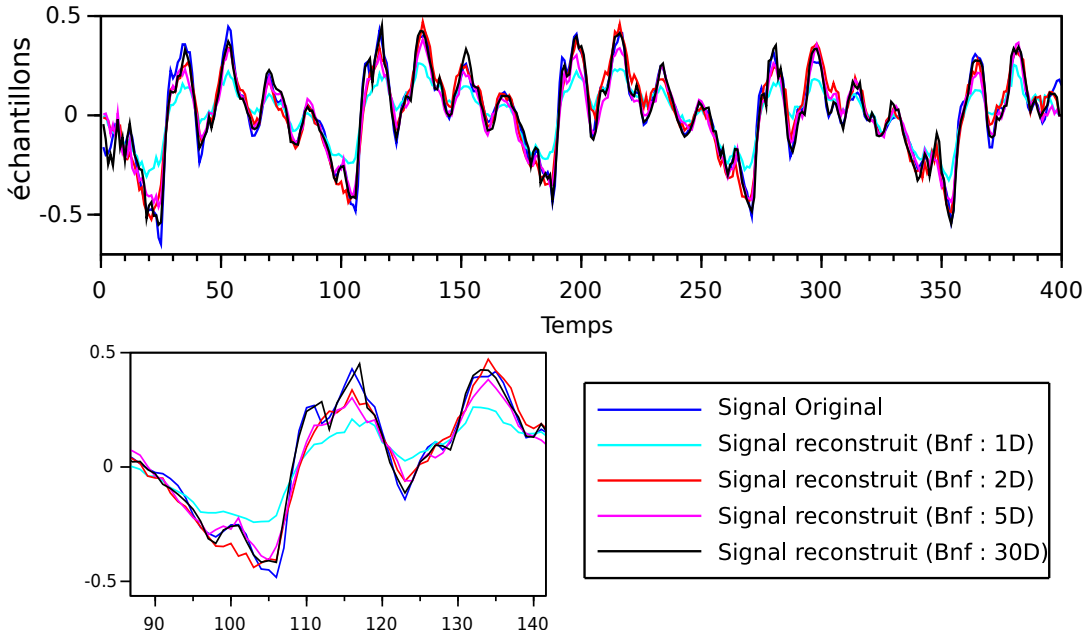

FIGURE 5.15 – Exemple de reconstruction d'un signal à l'aide d'un AE selon la taille de la couche de Bottleneck (1, 2, 5 ou 30)

Nous avons évalué les résultats selon le nombre de neurones de la couche centrale à l'aide de l'erreur RMS et affichés les résultats dans la figure 5.16. À partir d'une vingtaine de neurones, la diminution de l'erreur avec l'augmentation du nombre de neurones devient plus faible.

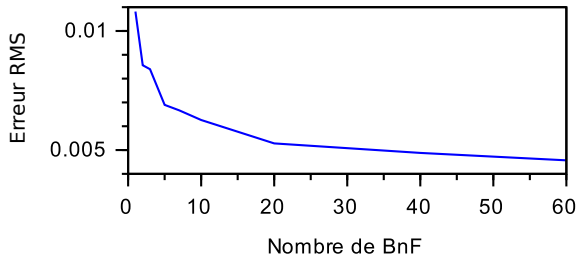

FIGURE 5.16 – Erreur RMS moyenne selon le nombre de neurones dans la couche centrale

Paramètres générés

Suite à différentes expériences, nous avons constaté qu'une reconstruction du signal de parole avec seulement deux neurones dans la couche de Bottleneck reste compréhensible. Les valeurs obtenues en sortie de cette couche contiennent donc une information suffisante pour que la parole soit reconnue par l'oreille humaine. Nous avons donc étudié l'utilité potentielle de ces paramètres en les visualisant.

La figure 5.17 représente les répartitions des valeurs des paramètres. Sur cet exemple, les valeurs sont toutes regroupées en un seul ensemble, comme nous le voyons sur les deux figures. Il n'y a donc pas plusieurs lots qui se démarquent pour aider par la suite à distinguer de manière non supervisée les emplacements des phonèmes, contrairement aux résultats obtenus avec les AE apprenant sur les bancs de filtres ou les MFCC, comme nous l'avions vu dans la figure 5.3.

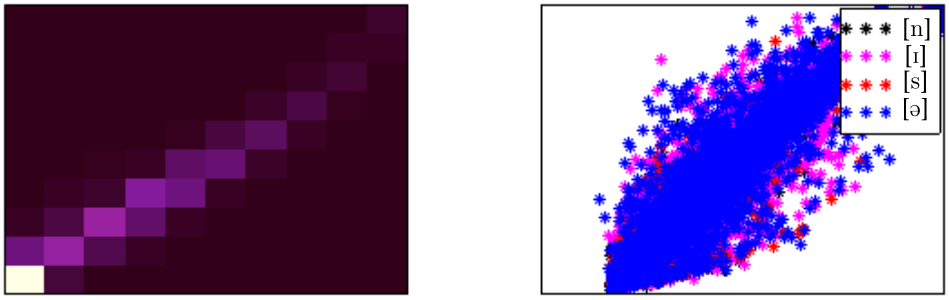

FIGURE 5.17 – Exemples de répartitions de paramètres de dimension 2, générés par un AE à partir du signal audio brut

Pour mieux estimer la pertinence de ces paramètres pour la discrimination des différents phonèmes, nous avons analysé les répartitions de quelques classes phonétiques éloignées (voyelles/consonnes/fricatives/...) comme nous l'avions fait en début de chapitre, et affiché les résultats dans la figure 5.17 de droite.

Le graphique résultant montre que les classes sont totalement mélangées, avec une répartition similaire pour les différentes classes phonétiques.

Nous avons essayé de comprendre pourquoi ces paramètres ne sont pas du tout adaptés à la reconnaissance phonétique malgré leur efficacité pour la compression de la parole. Nous avons donc étudié un cas extrême : utiliser sur le signal brut un AE convolutionnel avec une couche centrale d'un seul neurone. La figure 5.18 illustre les reconstructions effectuées par le réseau en fonction de la valeur du neurone, pour toutes les valeurs prises sur 4 plages de valeurs : [0,0556571 ; 0,0556642], [0,0808311 ; 0,0808474], [0,0725052 ; 0,1725695] et [0,3220179 ; 0,3220640]. De très faibles variations de la valeur d'activation du neurone central (moins de 0,1% de différence entre les différentes valeurs de chaque sous-ensemble) entraînent des changements radicaux dans le signal généré (consonne, voyelle, bruit...) comme le montre la figure 5.18.

5.2.5 Conclusion

Les réseaux de neurones de type Auto-Encodeur semblent bien adaptés à des tâches de compression de la parole. Cependant, les paramètres obtenus ne sont pas forcément utilisables pour de la classification non-supervisée.

Pour évaluer les nouvelles représentations paramétriques, nous avons utilisé un petit réseau de neurones dense d'une couche cachée de 100 neurones, entraîné au décodage phonétique. La pertinence des paramètres testés est jugée à l'aide du taux de bonne classification phonétique obtenue par ce réseau supervisé.

Les paramètres de l'AE ont été difficiles à optimiser, principalement à cause de la sensibilité du taux d'apprentissage à chaque nouveau changement du réseau. Finalement, au sens de notre mesure, ce sont les représentations générées à partir des

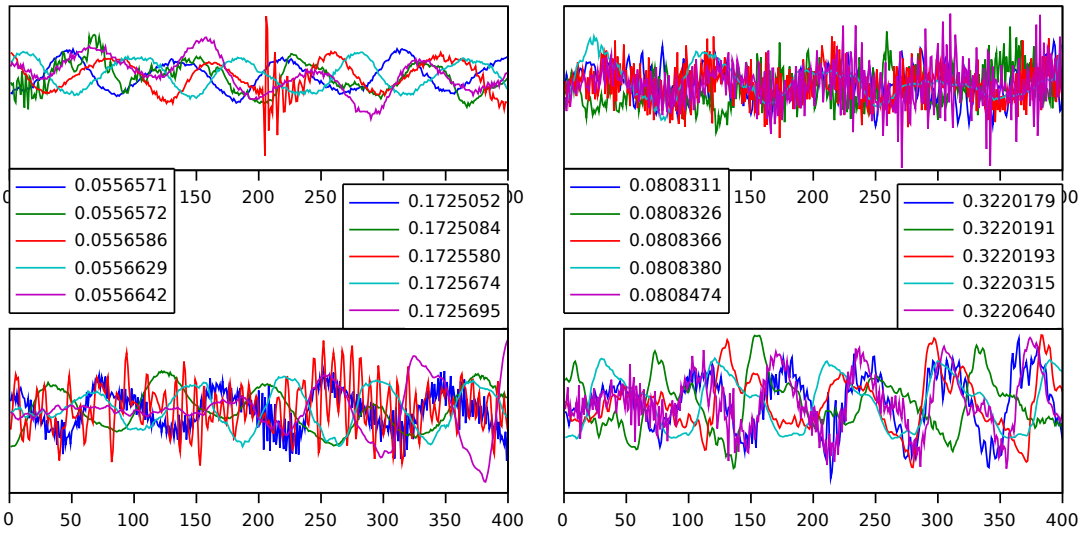

FIGURE 5.18 – Exemples de trames de signal reconstruites par l'AE selon la valeur d'activation du neurone central pour quatre lots de valeurs consécutives

paramètre de plus haut niveau (MFCC) qui ont été les plus discriminantes pour la reconnaissance phonétique et ont permis d'obtenir 41% de taux de classification correcte, c'est-à-dire 10% de mieux qu'une trame de MFCC non modifiée. Les bancs de filtres ont obtenus des résultats environ 10% moins bons. Il existe des pistes à suivre pour essayer de rendre efficaces les AE pour des tâches de regroupements, par exemple en ajoutant des contraintes sur les poids. Ainsi, nous pouvons, par exemple, utiliser des AE parcimonieux (sparse AE, sAE) (TANIGUCHI et al. 2016) qui conservent seulement les plus grandes valeurs en sortie de la couche générant les paramètres. Notons que le signal de parole a généré des paramètres efficacement compressés mais inadaptés à la reconnaissance phonétique et incompatibles avec une application non supervisée.

Après avoir expérimenté la génération de nouvelles représentations paramétriques issues des activations de la couche centrale d'un AE, nous avons changé d'approche et nous nous sommes intéressés à l'utilisation des valeurs des poids des neurones comme paramètres.

5.3 Extraction de LPC avec réseaux de neurones

5.3.1 LPC

Les coefficients LPC (JONES et al. 2009) peuvent être utilisés pour l'analyse et la synthèse de la parole. Ils décrivent l'enveloppe spectrale, mais se calculent dans le domaine temporel. Ce sont les coefficients du filtre linéaire auto-régressif minimisant l'erreur quadratique entre le signal original et le signal reconstruit à l'aide d'un bruit blanc gaussien filtré.

En pratique, le produit scalaire de N paramètres LPC avec N échantillons d'un signal est une prédiction de la valeur suivante prise par le signal. Nous pouvons apprendre à un réseau de neurones à effectuer la même tâche, c'est-à-dire, à partir des N précédents échantillons, prédire l'échantillon suivant. Si le réseau est composé uniquement d'une couche de sortie linéaire, alors ses poids seront des coefficients effectuant la même tâche que des coefficients LPC. La principale différence vient de la

méthode de calcul des poids : le réseau de neurones corrige ses poids en fonction de l'erreur de prédiction pour converger vers la solution idéale alors que les coefficients LPC sont habituellement obtenus en résolvant une équation matricielle.

5.3.2 Imitation des paramètres LPC : prédiction par réseau de neurones

Comme dit précédemment, nous voulons que le réseau prédise la valeur suivante d'un signal à partir de ses valeurs précédentes. Le schéma est illustré dans la figure 5.19. C'est un réseau sans couche cachée, avec seulement une couche de sortie composée d'un unique neurone. Nous entraînons le réseau sur les mêmes données que lors d'une paramétrisation LPC : un réseau par trame de parole, dont les valeurs des poids du neurone de sortie sont les pseudo coefficients LPC.

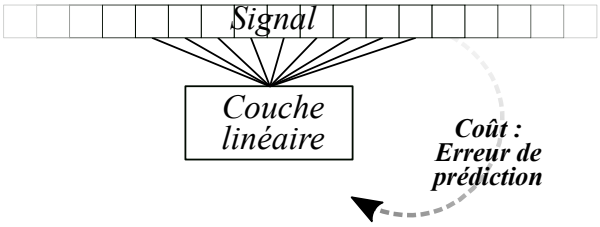

FIGURE 5.19 – Structure du réseau de neurones imitant la création de paramètres LPC

Le réseau étant minimal et la tâche simple, le réseau apprend bien. La faible quantité de données (1 trame, soit 400 valeurs pour des fenêtres de 25 ms d'un signal de 16 000 échantillons par seconde) nécessite néanmoins de très nombreuses itérations (plusieurs milliers) avant que le coût ne se stabilise. L'erreur de reconstruction, dont un exemple est affiché figure 5.20, est similaire à une erreur de reconstruction issue d'une véritable paramétrisation LPC, avec du bruit ajouté à un signal ressemblant à un peigne de Dirac (pics avec espacement régulier).

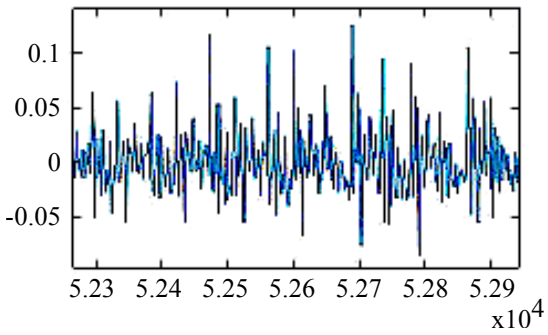

FIGURE 5.20 – Erreur de reconstruction du réseau de neurones imitant la création de paramètres LPC

La figure 5.21 nous montre, sur un extrait du signal original, la reconstruction issue d'une vraie paramétrisation LPC et celle issue de notre réseau de neurones. Le signal prédit par notre réseau semble légèrement moins bon : il accentue davantage les formes du signal avec parfois des sommets plus marqués.

Pour analyser l'origine de cette différence avec les coefficients LPC, nous avons regardé les différences de propriétés des paramètres estimés. Nos coefficients ont une réponse impulsionnelle plus forte, c'est-à-dire que le filtrage d'un Dirac augmente au lieu d'être stable ou de diminuer comme le font les coefficients LPC, dont les racines du polynôme formé par ses coefficients se situent à l'intérieur du cercle

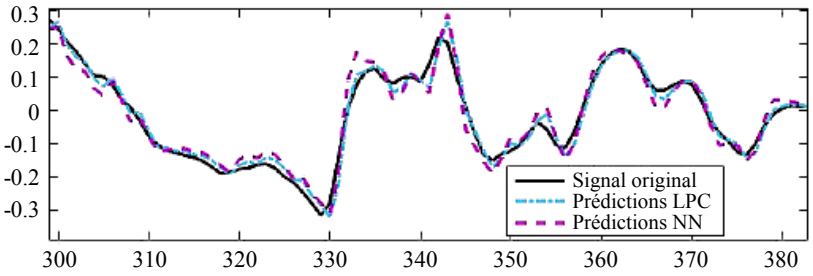

FIGURE 5.21 – Signal original et signaux reconstruits à l'aide de vrais coefficients LPC et ceux générés par notre réseau de neurones

unitaire. Nous avons affiché dans la figure 5.22 deux exemples de réponse impulsionnelle, issue des coefficients LPC et les nôtres estimés sur un même segment de parole, dans le graphique de gauche. L'intensité de la réponse des coefficients LPC (en bleu) diminue rapidement mais celle des coefficients de notre réseau de neurones explose. Dans le graphique de droite, nous avons affiché les racines des polynômes formés par les coefficients des deux méthodes. Nous voyons dans cet exemple que les racines issues des coefficients de notre réseau de neurones sont effectivement à l'extérieur du cercle, contrairement à celles issues des véritables coefficients LPC.

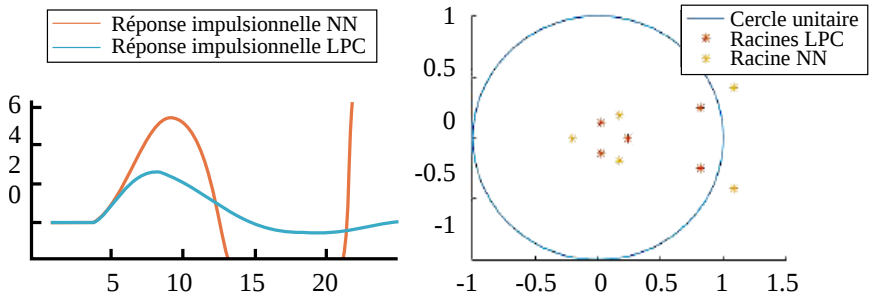

FIGURE 5.22 – Réponses impulsionnelles des coefficients LPC et des coefficients issus du réseau de neurones (*à gauche*) et positionnement dans le plan complexe des racines du polynôme formé par les coefficients (*à droite*)

Un exemple d'évolution au cours du temps des paramètres issus du réseau est montré dans le spectrogramme de la figure 5.23. Nous remarquons une grande fluctuation des ordres de grandeur des valeurs malgré l'ajout d'une contrainte sur la norme de type l2 pour diminuer ces différences. Ces différences rendent les paramètres obtenus beaucoup plus difficiles à utiliser par la suite. Les frontières phonétiques sont affichées en lignes pointillées noires. Certaines augmentations de l'intensité des paramètres semblent être liés à des segments phonétiques.

Pour évaluer l'utilité que ces paramètres peuvent avoir, nous avons regardé les moyennes et variances des représentants de différents phonèmes. La figure 5.24 montre des différences marquées dans les moyennes (figure du haut), sauf pour [m] et [n] qui donnent des valeurs moyennes très proches. Mais le graphique du bas montre que les variations (écart-type) sont du même ordre de grandeur que les moyennes pour presque une moitié des phonèmes.

Ces paramètres ne sont pas encore suffisamment bons pour séparer les phonèmes. Pour l'instant, obtenir ces pseudo paramètres LPC par réseau de neurones n'a pas d'avantage en terme de qualité des résultats. Le seul intérêt que cela peut avoir c'est en cas de calcul d'un grand nombre de coefficients LPC, par exemple pour étudier un écho grâce à eux. Le réseau de neurones apprend facilement alors

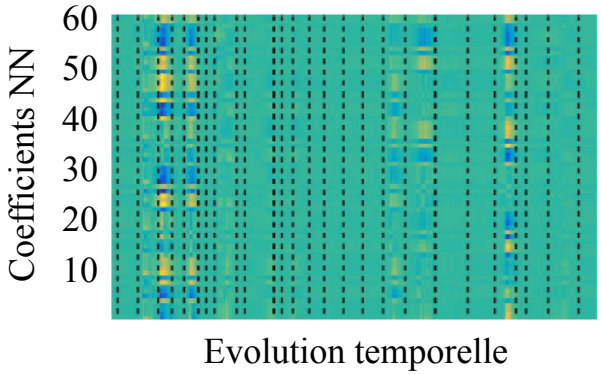

FIGURE 5.23 – « Spectrogramme » représentant l'évolution des paramètres LPC au cours du temps, avec les frontières phonétiques superposées (en noir)

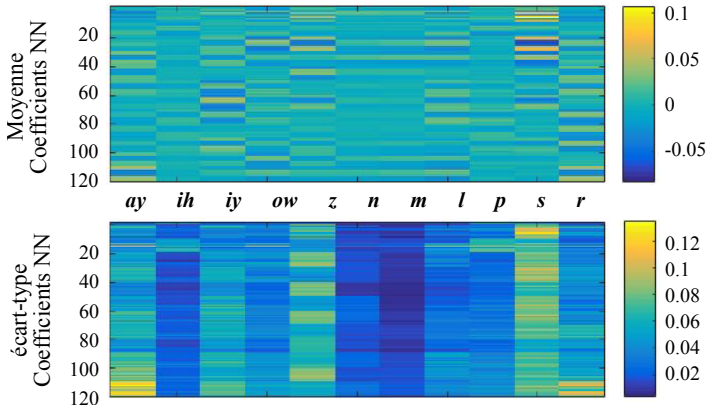

FIGURE 5.24 – Moyennes et écart-types des pseudo-paramètres LPC extraits pour 11 classes phonétiques différentes

que le calcul des vrais LPC devient extrêmement long pour un grand nombre de coefficients.

5.3.3 Prédictions d'un réseau de plusieurs couches

Même si les résultats préliminaires précédents n'ont pas été probants, nous avons continué nos expériences exploratoires en nous focalisant sur les possibilités offertes par les réseaux de neurones. Augmenter le nombre de coefficients n'est pas une solution utile : les premiers coefficients sont les plus importants car les plus proches de la valeur à prédire. Pour mieux saisir l'information apportée par des valeurs plus éloignées et davantage utiliser les possibilités des réseaux de neurones, nous avons donc tenté de construire un réseau plus profond, avec au moins une couche cachée, et des fonctions d'activations non linéaires.

La diminution du coût selon la taille du réseau, illustré en figure 5.25, montre que le réseau est bien plus efficace s'il possède au moins une couche cachée. Cela donne de l'espoir quant à l'utilité potentielle de pseudo paramètres LPC issus d'un réseau profond. Néanmoins une architecture « profonde » (plus d'une couche) pose des difficultés.

Une difficulté est visible dans la figure 5.26, qui illustre un réseau à une couche cachée et toujours avec une couche de sortie à un seul neurone. Le premier problème qui apparaît est que les neurones de la couche cachée ne sont pas ordonnés

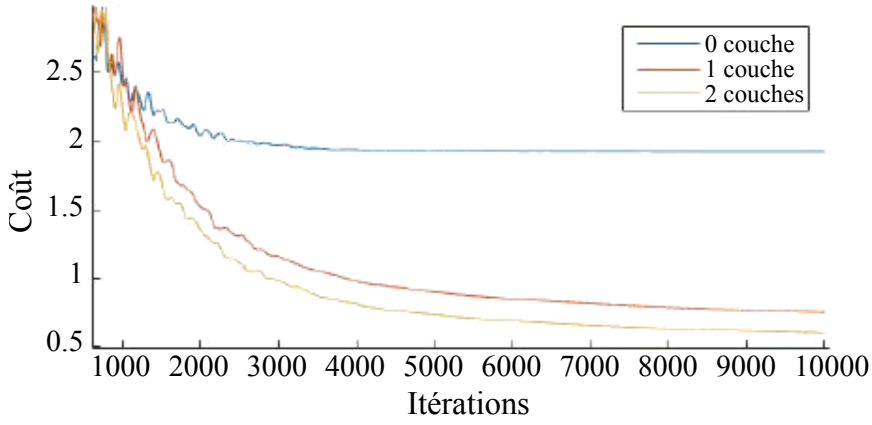

FIGURE 5.25 – Diminution du coût en fonction du nombre de couches cachées

et peuvent être interchangés d'un apprentissage à un autre, même pour les mêmes données, selon l'initialisation aléatoire et l'ordre des données envoyées. Avoir un ordre des neurones différent (et donc des paramètres extraits) pour chaque segment ne convient pas comme vecteur paramétrique.

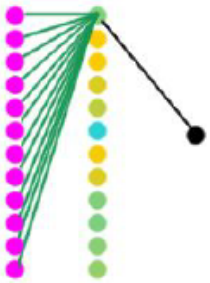

FIGURE 5.26 – Structure d'un réseau à une couche cachée et couche de sortie à un neurone

Face à ce problème, nous avons essayé plusieurs solutions :

— Utiliser les sorties des couches et non pas leurs poids ne s'apparente plus à estimer des coefficients de prédiction mais à utiliser la sortie d'une couche d'un auto-encodeur particulier, qui utilise le signal brut et prédit non pas les entrées données mais l'entrée suivante. Ces valeurs dépendent fortement du bruit et de la phase du signal et ne nous sont pas utiles.

— Apprendre le réseau sur l'ensemble des données puis ne réapprendre que les coefficients du neurone de sortie pour chaque segment. L'entrée de la dernière couche étant fixe, les poids de la couche de sortie gardent le même ordre d'un apprentissage à l'autre. Mais cela non plus ne nous a pas donné de résultats satisfaisants. Cependant, cette idée reste une piste intéressante à creuser.

— Réaliser un réseau qui effectue des regroupements plutôt que de générer de nouveaux paramètres. Pour cela nous avons mis 30 neurones sur la couche de sortie (le nombre de groupes souhaités) et fait remonter les erreurs seulement par le neurone de sortie le plus proche de la sortie attendue pour peu à peu le spécialiser dans la classe correspondante. Contrairement à nos attentes, ce réseau s'est avéré difficile à entraîner car il y a, grâce au hasard, toujours une sortie très proche de la sortie souhaitée sur les 30 valeurs potentielles générées. Le choix du neurone de sortie pour l'apprentissage était

donc notamment dû au hasard et, de plus, il n'y avait pas beaucoup d'erreur à faire remonter dans les couches. Nous avons réalisé quelques expériences pour remédier à ce problème, comme sélectionner comme neurone de sortie celui donnant le moins d'erreur sur la trame entière. Mais pour estimer l'utilité potentielle de cette piste, il est plus judicieux d'utiliser de la supervision et de s'aider de la classe phonétique réelle des segments pour sélectionner le neurone correspondant à la classe phonétique de chaque échantillon. Mais comme nous le constatons sur la figure 5.27, l'erreur issue d'un réseau est très variable, qu'il soit appris sur l'ensemble des données (en bleu) ou seulement sur les données de même classe phonétique (en orange). Même si l'erreur issue du réseau appris avec supervision est généralement moindre, la différence ne s'est pas avérée suffisante durant nos expériences préliminaires. La possibilité d'un réseau de neurones non supervisé effectuant des regroupements reste néanmoins une finalité suffisamment importante pour motiver de continuer des recherches allant dans ce sens.

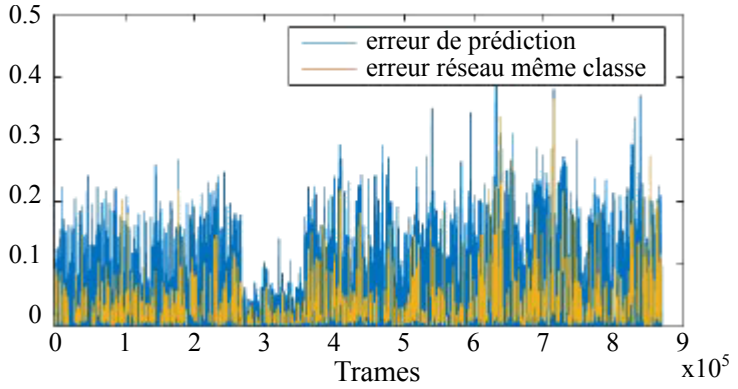

FIGURE 5.27 – Erreurs de prédiction d'un réseau appris sur toutes les données ou seulement sur les données de la même classe phonétique

À l'heure actuelle, nous n'avons pas obtenu de résultats satisfaisants sur ces expériences préliminaires sur les pseudo-paramètres LPC mais nous gardons en tête plusieurs pistes à suivre pour un travail futur.

5.4 Conclusion

En théorie, l'Auto-Encodeur semble adapté à notre tâche et c'est vrai en pratique dans certaines conditions. Ainsi, l'utilisation en entrée de plusieurs trames voisines de paramètres acoustiques déjà pertinents tels que les MFCC nous a permis de compresser l'information issue du voisinage et d'obtenir des paramètres encore plus efficaces (+7%). Les bancs de filtres, trop bas niveau, n'ont quant à eux pas permis d'obtenir de bons résultats.

L'Auto-Encodeur a prouvé qu'il pouvait être performant pour la tâche explicitement demandée, c'est-à-dire synthétiser un signal audio à partir d'un vecteur de paramètres de (très) petite taille. Cependant, les paramètres que nous avons obtenus étaient beaucoup trop compressés pour avoir un intérêt dans notre tâche de découverte de phonèmes.

Enfin, notre dernière expérience en matière de réseaux de neurones non supervisés a été d'imiter une paramétrisation LPC en prédisant les valeurs suivantes du signal et gardant les poids des réseaux appris sur de petits segments de parole comme paramètres. Cette expérience assez originale reste pour l'instant en suspens, faute

de résultats probants. Malgré tout, des améliorations sont possibles. Par exemple, nous pouvons espérer obtenir une réponse impulsionnelle stable ou décroissante en modifiant le calcul du coût. Même si nous ne pouvons pas directement forcer les racines du polynôme formé par les poids de notre réseau de neurones à être dans le cercle unitaire, nous pouvons ajouter une contrainte dans le coût d'apprentissage pour que la prédiction soit d'intensité inférieure à la valeur à prédire et donc que la réponse impulsionnelle n'explose pas.

Chapitre 6

Classification non supervisée en phones

Sommaire

6.1 Introduction

Les corpora de parole annotés abondent pour les langues les plus parlées, mais la grande majorité des langues ou des dialectes sont peu dotés en corpus annotés manuellement. Pour surmonter ce problème, la découverte non supervisée de pseudo-unités linguistiques dans la parole continue prend de l'ampleur ces dernières années, encouragée par exemple par des initiatives telles que le *Zero Resource Speech Challenge* (ZRSC), organisé en 2015 et 2017 (VERSTEEGH, THIOLLIERE et al. 2015 ; DUNBAR et al. 2017), comme nous l'avons vu dans l'état de l'art 2. Lors de ces deux éditions, la première tâche du challenge consistait à générer de nouveaux paramètres discriminants au niveau phonétique pour permettre de différencier les différents phonèmes.

Nous nous sommes intéressés à la découverte automatique des pseudo-unités dans la parole au niveau du phonème (pseudo-phones) sans supervision. Les pseudo-phones sont des sous-unités lexicales qui peuvent être plus courtes ou plus longues que les vrais phonèmes. Dans ce contexte, nous nous intéressons aussi à la découverte de nouvelles représentations paramétriques ayant des capacités discriminantes pour les classes phonétiques, une des deux sous-tâches du ZRSC.

Dans le cadre du challenge ZRSC 2017, nous avons soumis une représentation paramétrique simple mais efficace : les distances euclidiennes entre les données et les centroïdes, obtenus avec les k-means utilisant des MFCC blanchis en entrée. Cette représentation a obtenu des scores meilleurs que la représentation de base du ZRSC (MFCC standards) de 3% à 5% absolus selon la métrique utilisée dans le challenge (PELLEGRINI, MANENTI et PINQUIER 2017).

L'approche que nous avons utilisée dans ce chapitre est d'alterner des étapes de regroupement avec des étapes de classification. Nous avons d'abord regroupé les segments de parole pour obtenir des pseudo-labels puis entraîné un réseau de neurones avec eux. Le réseau de neurones a ensuite servi à générer de nouvelles représentations paramétriques : les activations de l'avant-dernière couche dense cachée. L'efficacité de notre approche réside dans la répétition de ces deux étapes plusieurs fois pour améliorer itérativement la qualité des regroupements et les capacités discriminantes des représentations.

Des approches itératives peuvent être trouvées dans la littérature (C.-T. CHUNG, TSAI, H.-H. LU et al. 2015 ; WANG, T. LEE, LEUNG, MA et al. 2015 ; HECK, SAKTI et NAKAMURA 2016a). Le travail le plus similaire au nôtre alterne l'identification des unités et sous-unités lexicales, à l'aide de DTW, HMM et DNN multi-tâches, et la génération d'une nouvelle représentation paramétrique, issue aussi d'une couche cachée du réseau de neurones (C.-T. CHUNG, TSAI, H.-H. LU et al. 2015). Les principales différences avec notre travail sont que nous utilisons des méthodes de regroupement pour identifier les pseudo-phones et que nous travaillons uniquement au niveau des sous-unités lexicales.

Parmi les méthodes de regroupement existantes, nous nous sommes intéressés aux k-means, aux GMM et aux DPGMM. Dans (WANG, T. LEE et LEUNG 2011), les k-means sont utilisés sur des paramètres générés par un Auto-Encodeur (AE) puis binarisés. Ils sont utilisés de manière similaire dans (TIAN et al. 2014), avec AE et regroupements par graphes. Les DPGMM ont été utilisés avec succès dans les deux premières éditions du ZRSC, obtenant les meilleurs résultats en 2015 (H. CHEN et al. 2015) et faisant partie notamment des deux meilleurs modèles en 2017 (HECK, SAKTI et NAKAMURA 2017 ; H. CHEN et al. 2017).

6.1.1 Plan

Dans ce chapitre, nous allons présenter notre système non supervisé de génération de nouveaux paramètres de parole discriminants au niveau phonétique et de découverte de sous-unités linguistiques.

Nous évaluons d'abord notre approche sur une tâche de découverte de pseudo-phones. Les résultats sont donnés en termes de pureté des regroupements. Ces expériences permettent de concevoir notre approche globale et de définir les hyperparamètres des modèles. Nous avons regardé l'utilité que ces pseudo-labels pouvaient avoir pour la découverte des mots. Ensuite, nous rapportons les résultats des représentations générées en termes de taux d'erreur ABx.

Pour nos premières expériences, nous utilisons trois corpora ayant des langues, des tailles et des conditions différentes : BUCKEYE, BREF et NCHLT. Ensuite, nos expériences sont réalisées sur les données du ZRSC 2017 : il s'agit des trois langues d'entraînement du challenge (anglais, français et mandarin).

6.2 Description du système

Comme le montre la figure 6.1, notre système est composé d'une méthode de regroupement (non supervisée) et d'une méthode de classification (supervisée). La méthode de regroupement (k-means, GMM ou DPGMM) regroupe les segments de parole en classes et celles-ci permettent l'entraînement de la méthode de classification (NN). Une fois entraîné, le réseau de neurones peut être utilisé pour générer de nouveaux paramètres (*posteriorgrams* ou *posteriorgrammes*) en extrayant la sortie d'une couche cachée.

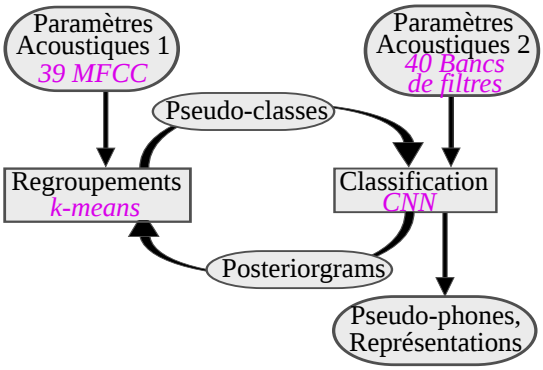

FIGURE 6.1 – Architecture générale : le système est entraîné de manière itérative. En mauve sont indiqués les paramètres et les méthodes nous ayant permis d'obtenir les meilleurs résultats

Lors de la première itération, les paramètres d'entrée utilisés par le k-means sont les paramètres standard : 13 MFCC et leurs deux premières dérivées. Le CNN est ensuite entraîné sur 40 bancs de filtres (désignés par « Paramètres acoustiques 2 » sur la figure) pour prédire les classes du k-means. Ses couches cachées contiennent les données d'entrée reformulées de manière à répondre à la tâche de classification en pseudo-phones. Les paramètres extraits d'une de ses couches cachées ont permis aux k-means d'être plus efficaces lorsqu'ils étaient ajoutés aux MFCC précédemment utilisés. Les k-means des itérations suivantes utilisent donc ces deux paramètres comme entrée. Ils obtiennent alors de meilleurs regroupements, en terme de pureté, qui permettent à leur tour de mieux entraîner le CNN.

6.2.1 Regroupement

Nous comparons 3 méthodes de regroupement (clustering, en anglais) : les k-means, les GMM et les DPGMM. Les DPGMM sont des GMM utilisant des processus de Dirichlet pour choisir automatiquement le nombre de clusters, en maximisant la vraisemblance de la répartition des données (H. CHEN et al. 2015). Il n'y a donc pas à choisir manuellement le nombre de clusters *a priori* comme pour le k-means, mais il y a un autre paramètre à ajuster pour estimer le nombre de groupes : le paramètre alpha. Celui-ci permet de privilégier ou désavantager les regroupements et influence donc le nombre de groupes finaux.

Parmi les paramètres d'entrée vus dans les chapitres précédents, ces méthodes peuvent utiliser les bancs de filtres ou les MFCC. Nos k-means utilisent la distance euclidienne. Quelques expériences ont été réalisées avec la distance cosinus (k-means sphérique) mais sans apporter d'amélioration notable aux résultats. Avant d'envoyer les paramètres d'entrée au k-means, nous avons remarqué qu'il était préférable d'utiliser la ZCA, une méthode de blanchiment présentée dans le chapitre 2. À notre connaissance, la ZCA n'est pas communément utilisée en traitement de la parole. C'est une transformation proche de la PCA qui a la particularité de conserver l'orientation des données.

Après le regroupement, nous obtenons des classes (pseudo-labels) et de nouvelles représentations paramétriques : le vecteur des distances avec chaque centroïde (pour le k-means) ou le vecteur de probabilité des pseudo-classes (pour le GMM et le DPGMM). Les pseudo-classes nous permettent d'entraîner le CNN de manière supervisée.

6.2.2 Classification : réseau de neurones

Nous utilisons un réseau de neurones pour la tâche de classification en unités sous-lexicales (pseudo-phones). Il est entraîné avec les pseudo-labels obtenus par la méthode de regroupement. Comme nous l'avons dit dans les chapitres précédents, les réseaux de neurones s'adaptent à toutes les entrées : bancs de filtres, MFCC ou même directement le signal.

L'architecture optimale du réseau dépend des paramètres d'entrée. Par exemple, un réseau uniquement composé de couches denses conviendra aux MFCC alors que le signal brut aura obligatoirement besoin de filtres temporels. Les bancs de filtres peuvent être utilisés en entrée d'un MLP mais donneront de moins bons résultats qu'un CNN, utilisant des filtres temporels et fréquentiels, qui sera plus adapté. De même, la taille et le nombre de filtres, le nombre de couches de filtres et de couches totalement connectées dépendront eux aussi de l'entrée. Pour chaque entrée possible, nous avons optimisé les différents paramètres du réseau sur une tâche d'apprentissage supervisé avec les vraies classes annotées manuellement, dont certains résultats sont donnés dans le chapitre 3.

La principale évolution du réseau est l'utilisation par les couches denses des sorties de plusieurs des couches précédentes. Un réseau similaire peut être trouvé dans la littérature sous le nom de DenseNet (G. HUANG et al. 2017). Construire un réseau de ce type peut nécessiter de nombreux ajustements : il faut trouver les meilleurs liens possibles entre les couches en plus des autres hyperparamètres. Pour alléger le travail d'ajustement, nous nous sommes appuyés sur le CNN construit dans le chapitre 3 et avons conservé les deux couches de convolution suivies des deux couches denses, comme montré dans la figure 6.2. Ce type de réseau n'a pas eu un impact

significatif sur les résultats supervisés mais a apporté une amélioration dans la génération des paramètres pour le k-means et a été conservé par la suite.

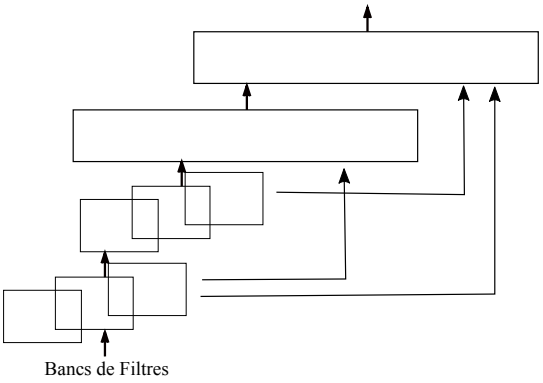

FIGURE 6.2 – Structure du réseau construit, avec deux couches de convolution suivies de deux couches denses. Chaque couche prend en entrée les sorties des couches précédentes.

Les fonctions d'activation sont les tangentes hyperboliques pour les couches de convolution et *ELU* pour les couches denses. Les couches denses ont 150 neurones et sont similaires aux couches d'un réseau de type denseNet (G. HUANG et al. 2017), c'est-à-dire que les couches prennent en entrée une concaténation des sorties de plusieurs des couches précédentes. La couche de sortie du réseau apprend à prédire les pseudo-labels donnés par l'étape de regroupement précédente : son nombre de neurones est donc égal au nombre de groupes formés. La fonction d'optimisation de mise à jour des poids peut être choisie parmi les fonctions suivantes : adam (KINGMA et BA 2015) ou adamax, adagrad (DUCHI, HAZAN et SINGER 2011), ou rmsprop (TIELEMAN et G. HINTON 2012) pour des résultats équivalents.

Tous les réseaux construits ont au plus deux couches de convolution et trois couches denses. L'architecture des réseaux selon l'entrée est celle-ci :

— Bancs de filtres
  - 2 couches de convolution avec filtres carrés (temps et fréquence) de taille 5x5, avec entre les deux une couche de max-pooling de taille 2x1.
  - 3 couches cachées denses.

— MFCC
  - 1 couche de convolution avec des filtres temporels (5x1).
  - 3 couches cachées denses.

Nous utilisons un taux d'apprentissage de 0,007 pour des ensembles d'apprentissage (mini-batch) de 128 exemples.

6.2.3 Métriques d'évaluation

Pureté

Pour évaluer les différentes configurations de notre système, nous nous appuyons sur la segmentation manuelle afin de calculer la pureté des regroupements en pseudo-phones. Cette métrique est définie dans l'équation 6.1 avec N le nombre de segments phonétiques, I le nombre de pseudo-labels, J le nombre de phonèmes et n_j^i le nombre de segments du phone j et automatiquement classé en tant que pseudo-phone i :

$$\text{Pureté} = \frac{1}{N} \sum_{i=1}^{K} \max_{j \in [1,C]} (n_j^i) \tag{6.1}$$

Taux d'erreur ABx

La représentation paramétrique générée est évaluée à l'aide du taux d'erreur ABx (SCHATZ, PEDDINTI, BACH et al. 2013 ; SCHATZ, PEDDINTI, CAO et al. 2014). La discriminabilité ABx entre les catégories x et y est définie comme la probabilité que les échantillons A et X soient plus éloignés qu'un échantillon B avec X (A et X étant de la catégorie x et B étant de la catégorie y). Ceci est fondé sur une distance d sur l'espace (dépendant du modèle) des représentations paramétriques pour ces sons. Étant donnés deux ensembles de représentations paramétriques que nous souhaitons évaluer, $S(x)$ et $S(y)$ respectivement de la catégorie x et y, la métrique estime cette probabilité en utilisant la formule suivante :

$$\frac{1}{m(m-1)n} \sum_{A \in S(x)} \sum_{B \in S(y)} \sum_{X \in S(x) \setminus \{A\}} (1_{d(A,X) > d(B,X)} + \frac{1}{2} 1_{d(A,X) == d(B,X)}) \tag{6.2}$$

avec $m = \#S(x)$ et $n = \#S(y)$.

Dans le contexte de ZRSC 2017, deux taux d'erreur ABx sont calculés pour chaque langue/durée : l'un calculé pour un même locuteur (la condition *within*) et l'autre entre différents locuteurs (la condition *across*). Plus l'ABx est petit, meilleures sont les représentations paramétriques.

6.3 Ajustement des paramètres du système

La première étape consiste à choisir les paramètres acoustiques et la méthode de regroupement. Ensuite, nous devons optimiser les hyperparamètres pour l'étape de classification, puis pour la boucle itérative. Pour ce faire, nous utilisons le corpus de français BREF80 et la pureté (des regroupements) comme métrique. Pour construire et valider notre approche et ajuster les hyperparamètres, nous travaillerons avec 30 regroupements, ce qui est proche du nombre actuel de monophones utilisés pour modéliser les phonèmes du français.

6.3.1 Regroupement

Comme nous l'avons vu dans l'état de l'art, les DPGMM ont récemment été utilisés dans les systèmes ayant obtenus les meilleurs scores au ZRSC. Nous avons donc fait une première expérience sur les paramètres de Bottleneck issus d'une couche de deux neurones centrale d'un Auto-Encodeur, comme vu dans le chapitre 5, pour 5 classes phonétiques différentes éloignées les unes des autres et un seul locuteur. La figure 6.3 affiche 4 graphiques issus des mêmes données, de gauche à droite :

— La répartition des points sous forme de (courbes de densité) qui permettent de voir trois zones denses principales se démarquer et quelques pics de densité plus faibles éparpillés autour.
— La répartition des points avec la couleur de chaque point dépendant de la vérité terrain. En apparence, les classes semblent avoir un faible recouvrement.
— La répartition des classes obtenues par un k-means. Le principal problème vient de la séparation des classes linéaire entre chaque centroïde et ne suivant pas les vraies formes des répartitions des individus. La partie pas tout à fait non supervisée et l'indication du nombre de groupes à trouver (ici 5).
— La répartition des classes obtenues par un DPGMM. Le DPGMM de scikit-learn (PEDREGOSA et al. 2011) utilise lui aussi un nombre de classe *a priori*.

Par contre, il est beaucoup plus efficace pour mieux suivre les contours des répartitions et se rapproche davantage de la réalité terrain.

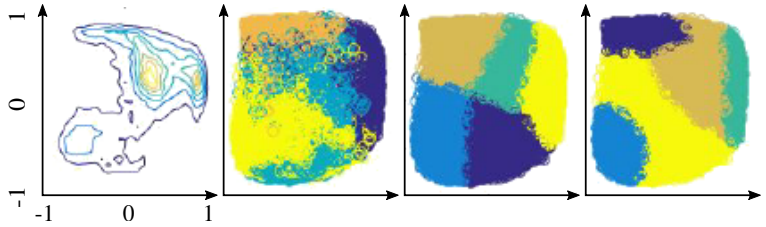

FIGURE 6.3 – Répartitions sur un plan 2D des exemples (*premier, à gauche*), selon leur véritable classe phonétique (*deuxième*), les regroupements par k-means (*troisième*) et les regroupements par DPGMM (*quatrième*)

Sur ces données, le DPGMM semble être plus efficace que le k-means. Mais, après une série d'expériences, cela n'a pas été le cas sur nos données de plus grande dimension.

Pour choisir les paramètres d'entrée et la méthode de regroupement, nous avons évalué la pureté des différents groupes obtenus sur les paramètres habituels et reporté les résultats dans la table 6.1. Pour cette étape de regroupement, une seule trame est utilisée en entrée, sans utilisation des trames voisines. Nous avons réalisé comme expérience l'ajout de deux trames de voisinage (une avant et une après la trame courante) et cela a légèrement diminué les performances. Ajouter du contexte semble principalement ajouter du bruit dans les données d'entrée et dans les distances, calculées sur de plus grands vecteurs. De plus, dans le cas des MFCC, l'utilisation des deux premières dérivées permet de déjà prendre en compte le contexte proche.

TABLE 6.1 – Pureté des groupes (pseudo-phones) (%) avec différents paramètres d'entrée et méthodes de regroupement sur BREF

Paramètres d'entrée	k-means	GMM	DPGMM
Bancs de filtres	30	32	31
Bancs de filtres avec ZCA	28	23	**33**
MFCC	34	33	32
MFCC avec ZCA	**40**	**34**	31

Un k-means sur les MFCC (et leurs deux premières dérivées) blanchis à l'aide d'une ZCA a obtenu les meilleurs résultats (40%). L'application d'une ZCA aux bancs de filtres a dégradé les résultats pour le k-means (-2%) et le GMM (-9%) mais a augmenté ceux du DPGMM (+2%). Comme le k-means, le GMM préfère les MFCC blanchis (34%). Le DPGMM a des préférences inverses aux deux autres méthodes : ce sont les bancs de filtres blanchis qui lui permettent d'obtenir de meilleurs résultats alors que la ZCA diminue ses scores sur les MFCC (-1%). Cette table nous montre l'importance que peuvent avoir les paramètres d'entrée sur les résultats (plus de 10% de différence pour le k-means et le GMM). Les paramètres testés ont moins d'impacts sur le DPGMM (2% d'écart) mais aucun ne lui a permis d'atteindre les meilleurs résultats.

Avant d'envoyer les paramètres d'entrée au k-means, nous avons remarqué qu'il pouvait être préférable d'appliquer un blanchiment à l'aide d'une ZCA (COATES et NG 2012). Comme vu dans le chapitre 2, la ZCA se base sur la PCA mais conserve la

taille et l'orientation des données. La conservation de l'orientation des données permet de calculer et d'appliquer la ZCA locuteur par locuteur et d'obtenir des transformations spécifiques à chaque fichier. Cela nous a permis d'obtenir des paramètres plus robustes au locuteur et c'est donc cette ZCA appliquée indépendamment sur chaque locuteur que nous avons utilisée.

Les meilleurs résultats obtenus au challenge de ces deux dernières éditions utilisent des DPGMM. Leur avantage est de ne pas avoir à fixer le nombre de groupes *a priori* : ils le fixent eux-même en utilisant la divergence de Dirichlet. Il y a néanmoins un paramètre à fixer : le seuil qui permet d'influencer les décisions de regroupement des groupes et donc leur nombre. Durant nos expériences, nous avons malheureusement constaté que le nombre de groupes finalement obtenus était très faible (parfois 1 seul groupe trouvé). Cela peut être dû aux données trop bruitées pour cette méthode. C'est pourquoi nous avons utilisé le DPGMM de sklearn (PEDREGOSA et al. 2011), qui permet de choisir le nombre de regroupements obtenus.

6.3.2 Classification

Pour comparer l'utilité de deux types de paramètres standards (coefficients des bancs de filtres, MFCC) et du signal brut, nous avons construit pour chaque un réseau adapté, comme présenté dans la section 6.2.2. Les résultats obtenus en classification supervisée à l'aide de l'annotation manuelle par ces réseaux sont donnés par la table 6.2. Leurs résultats sont similaires (environ 60% de pureté) mais ils n'ont pas tous besoin du même temps d'apprentissage. Le réseau prenant le signal brut en entrée est plus gros et profond, puisqu'il a plus de transformations à effectuer, il est donc beaucoup plus long à entraîner. Nous avons choisi d'utiliser les bancs de filtres en entrée du réseau car ils ont donnés les meilleurs résultats.

TABLE 6.2 – Comparaison des taux de classification, obtenus à l'aide de réseaux de neurones et d'un apprentissage supervisé, selon les paramètres d'entrée choisis

Paramètres	BREF80 (Fr)
Signal brut	58
Bancs de filtres	62
MFCC	61

Comme nous l'avons fait pour les méthodes de regroupement, nous avons regardé l'impact sur les résultats de l'utilisation d'une ZCA sur les paramètres d'entrée. L'évolution de la pureté sur un ensemble de developpement durant l'apprentissage a été reportée sur le graphique 6.4. Nous voyons que la convergence du réseau est beaucoup plus rapide avec la ZCA que sans, mais la pureté finale est un peu moins bonne. Nous n'avons donc pas appliqué de ZCA sur les paramètres d'entrée du réseau de neurones.

Le dernier paramètre à optimiser est la durée du voisinage considéré. Entre 80 et 200 ms, la taille du contexte n'a pas d'influence significative sur les résultat. Nous avons choisi un voisinage de 130 ms.

6.3.3 Boucle itérative

Maintenant que nous avons optimisé le regroupement et la classification indépendamment l'un de l'autre, nous nous intéressons à leur interaction au sein de la boucle qu'ils forment. Nous devons choisir :

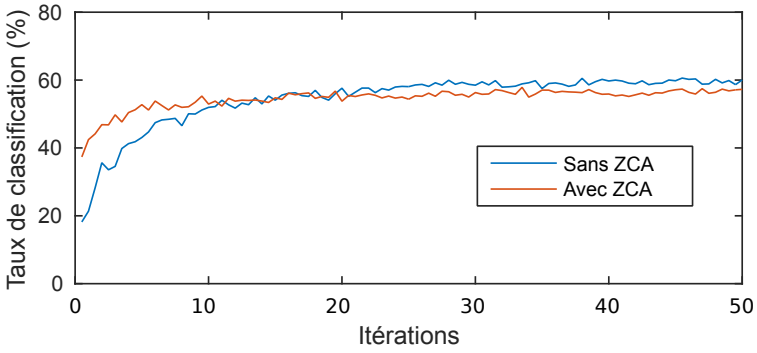

FIGURE 6.4 – Influence du blanchiment des bancs de filtres par ZCA sur l'apprentissage du réseau

— le(s) couche(s) du CNN qui génèrent les paramètres les plus efficaces,
— la méthode de regroupement la plus adaptée à ces nouveaux paramètres,
— le critère d'arrêt de la boucle,
— les sorties de la boucle (pseudo-labels et paramètres).

Les posteriorgrammes pris comme paramètres d'entrée par le k-means sont la sortie d'une des couches cachées du CNN. Pour choisir de quelle couche les paramètres seront issus, nous avons appliqué un k-means sur chacune des couches et comparé la pureté des regroupements obtenus. C'est avec l'avant-dernière couche (la dernière couche cachée) que nous avons obtenus les meilleurs résultats.

La couche de sortie, celle qui prédit les probabilités des différents groupes, obtient de mauvais résultats car elle utilise la fonction d'activation softmax qui génère des paramètres proches de vecteurs binaires : leurs valeurs sont proches de 1 pour la (ou les) classe(s) les plus probables et très proches de 0 pour les autres. D'autres travaux ont eux aussi obtenus les meilleurs paramètres parmi les dernières couches cachées (H. CHEN et al. 2017).

La table 6.3 donne les résultats de méthodes de regroupement (k-means, GMM et DPGMM) selon les paramètres d'entrée. Quels que soient les paramètres utilisés, les meilleurs regroupements ont été obtenus par le k-means. En concaténant les posteriorgrammes du CNN avec les MFCC blanchis par ZCA, il a obtenu 41% de pureté. Entraîner de nouveau un CNN sur les nouveaux groupes obtenus a permis d'améliorer la pureté de 1% absolu.

TABLE 6.3 – Résultats des regroupements sur BREF80 (Fr) (Pureté, %)

Entrée	k-means	GMM	DPGMM
ZCA-MFCC	40	34	31
CNN posteriorgrammes	40	30	34
… avec ZCA	38	38	32
… avec ZCA + ZCA-MFCC	**41**	38	33

Le critère d'arrêt de la boucle peut par exemple se fonder sur l'évolution des classes trouvées par le k-means ou celle prédites par le CNN. Pour plus de simplicité, dans ce travail, nous avons choisi de fixer le maximum d'itérations à 20.

En sortie de notre modèle, nous avons des pseudo-phones et de nouveaux paramètres. Les pseudo-phones, tout comme les paramètres, peuvent être ceux du k-means ou ceux du CNN : à la fin de la convergence, les classes du k-means et

celles du CNN nous ont donné une pureté similaire et nous avons choisi d'utiliser les sorties issues du k-means, dont les paramètres générés permettent d'obtenir un meilleur taux d'erreur ABx.

Les paramètres générés sont utilisés dans un cadre différent : la discrimination phonétique de la tâche 1 de l'édition 2017 du Zero Resource Speech Challenge. Nous avons donc comparé les différentes représentations générées à l'aide du taux d'erreur ABx utilisé dans ce challenge.

Le seul paramètre que nous ne pouvons pas optimiser à l'aide de la pureté obtenue en résultat est le nombre de regroupements. En effet, plus il y a de groupes, meilleure est la pureté, contrairement au taux d'erreur ABx qui compare les paramètres générés par les regroupements et non pas les regroupements eux-mêmes. La figure 6.5 nous montre l'influence du nombre de groupes. Le nombre de moyennes idéal du k-means diffère pour le *within* (intra-locuteur) et pour le *across* (inter-locuteur). Pour l'intra-locuteur, la diminution est importante jusqu'à 70 clusters puis faible jusqu'à 160 avant de stagner. L'inter-locuteur a un taux d'erreur ABx minimum aux environ de 30 à 100 regroupements. Les moins bons résultats pour un nombre de regroupements plus élevé peut peut-être s'expliquer par une spécialisation au locuteur des groupes. Nous avons décidé d'utiliser 160 groupes.

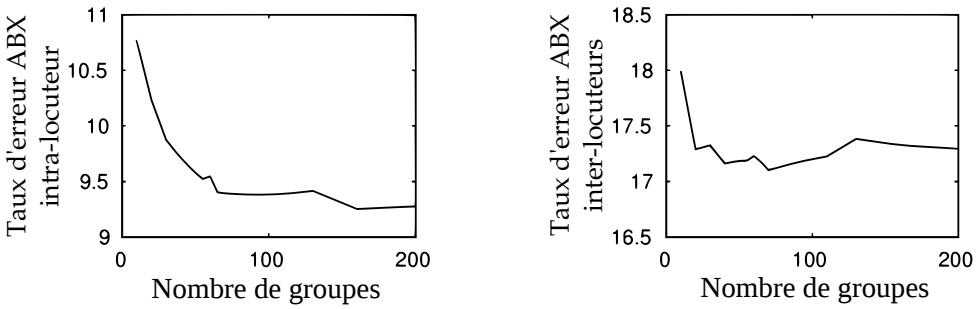

FIGURE 6.5 – Taux d'erreur ABx pour conditions intra-locuteur (*à gauche*) et inter-locuteurs (*à droite*) selon le nombre de groupes

6.4 Résultats au niveau lexical et sous-lexical

La allons maintenant regarder la pureté des regroupements obtenus et étudier rapidement leur potentielle utilité pour la découverte de pseudo-mots.

6.4.1 Pureté des regroupements

Il est intéressant de comparer les résultats avec ceux obtenus avec un apprentissage supervisé. Pour cela, nous utilisons la structure de CNN construit dans le chapitre 3 La table 6.4 nous donne les informations suivantes pour évaluer la qualité des résultats :

— Le taux de classification obtenu en classification supervisée utilisant l'annotation manuelle par le même modèle CNN que celui que nous utilisons en non supervisé. Nous n'avons pas cherché à construire un modèle compliqué pour augmenter au maximum les scores mais plutôt un modèle adapté à notre problème non supervisé. En comparaison, l'état de l'art obtient environ 80% de taux de reconnaissance de phonèmes sur le corpus TIMIT (BADINO 2016).

— Le pourcentage de pureté obtenu par notre modèle non supervisé (voir figure 6.1, après les 20 itérations (voir section 6.3.3). Les scores sont calculés

pour 30 pseudo-phones. Nous voyons qu'il y a presque 20% d'écart entre le petit corpus de xitsonga lu et le grand corpus d'anglais spontané.

TABLE 6.4 – Résultats (Taux de Classification (T_c) (%) et pureté (%)) pour trois langues : anglais (En, BUCKEYE), français (Fr, BREF80) et xitsonga (Xi, NCHLT) pour un CNN entraîné avec les vrais phonèmes annotés manuellement et les groupes issus de notre modèle itératif

Langue	En	Fr	Xi
T_c (%) Apprentissage supervisé	60	62	66
Pureté (%) pour 30 groupes	29	43	46

Ce sont les corpora de français et de xitsonga qui obtiennent les meilleurs résultats, en supervisé comme en non supervisé. Cela s'explique facilement : ce sont les deux corpora de parole lue, les plus petits et avec le moins de locuteurs différents. Ces trois critères influent beaucoup sur les résultats. Il est intéressant de noter que nous obtenons des résultats presque équivalents avec l'anglais si nous ne prenons que 30 minutes et un seul locuteur.

En étudiant en détails la composition des regroupements, nous avons constaté qu'obtenir 30% (respectivement 40%) de pureté ne signifiaient pas 70% (respectivement 60%) d'erreurs dues à des phonèmes totalement différents du label phonétique attribué au groupement. Les regroupements sont généralement constitués de 2 ou 3 lots d'exemples appartenant à des classes phonétiques proches. Ainsi, les phonèmes des trois principales classes phonétiques présentes dans chaque groupe représentent en moyenne 70% des exemples du groupe pour le français ou le xitsonga et 57% pour l'anglais.

6.4.2 Étiquettes des groupes proches

Nous avons analysé les résultats obtenus par le k-means. Le nombre de regroupements obtenus n'est pas obligatoirement proche du nombre de classes phonétiques du langage. Nous pouvons accepter que plusieurs groupes correspondent au même phonème, le travail de supervision pour attribuer une étiquette à un groupe étant minime. Plus il y a de groupes, moins il y a d'exemples par groupe et plus ces groupes sont purs. Nous avons donc commencé à regarder les résultats avec un nombre élevé de regroupements.

Sur un test utilisant 920 regroupements, nous avons regardé les classes phonétiques majoritaires dans chacun des 5 groupes les plus proches de l'exemple considéré dans la figure 6.6. Nous voyons que le groupe le plus proche (deuxième ligne) correspond, environ pour moitié, à un groupe de même classe que l'exemple considéré (première ligne). Sur cet exemple d'une trentaine de phonèmes, un bon groupe se trouve toujours parmi les 5 groupes les plus proches (sauf pour le « t » ici).

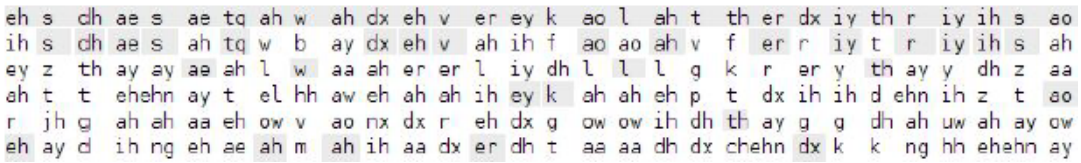

FIGURE 6.6 – Affichage des résultats pour un extrait avec la vérité terrain sur la première ligne et les étiquettes des 5 groupes les plus proches dans l'ordre en-dessous, avec en gris les étiquettes justes

La table 6.5 montre les pourcentage d'exemples ayant au moins un groupe de la bonne classe phonétique parmi les 1, 2, 3, 4 ou 5 groupes les plus proches. Nous voyons que doubler le nombre de regroupements permet généralement d'augmenter la pureté de 2%. La probabilité d'être bien regroupé varie de 21% (pour seulement 8 groupes) à près de 40% pour près de 1000 groupes. Si le groupe le plus proche n'est pas le bon, il y a en moyenne 40% de chances qu'un des 4 autres groupes les plus proches (du deuxième au cinquième) corresponde. Nous atteignons ainsi 80% de chance qu'un des 5 groupes les plus proches, parmi 920 groupes trouvés, corresponde à la bonne classe phonétique.

TABLE 6.5 – Taux de segments phonétiques (%) pour lesquels au moins 1 des n groupes les plus proches est majoritairement composé de segments de la même classe phonétique, avec n allant de 1 à 5

Nb de groupes utilisés	Nb de groupes proches 1	2	3	4	5
8	21	35	44	52	58
15	24	39	51	65	60
30	26	42	53	61	67
60	28	45	56	64	69
120	31	48	59	67	73
240	33	51	62	70	75
460	35	53	64	72	78
920	38	57	68	75	80

6.4.3 Utilité des regroupements pour un travail sur les mots

Nous avons réalisé des tests préliminaires pour évaluer l'intérêt que pourraient avoir ces regroupements dans la découverte non supervisée des mots d'une langue.

Pour ce faire, nous avons regardé la ressemblance entre les écritures phonétiques et les écritures en pseudo-phones des réalisations de mots. La table 6.6 permet de comparer à gauche des exemples de décompositions phonétiques du mot « *something* » et à droite les indices des groupes qui ont été attribués par notre système (par vote majoritaire sur chaque segment phonétique). La variabilité des phonèmes de ce mot est faible (les trois premiers phonèmes sont toujours les mêmes sur ces exemples). Il y a davantage de variabilité dans les attributions des groupes même si pour ces trois premiers phonèmes il y a un groupe majoritaire bien marqué (le n^o24, le n^o7 puis le n^o3) et deux ou trois groupes présents quelques fois. Les phonèmes de fin sont répartis sur davantage de groupes, particulièrement le I.

De manière plus générale, les phonèmes de la vérité terrain sont égaux dans 72% des cas alors que les pseudo-phones ont un score presque moitié moins élevé (35%), comme le montre les deux histogrammes affichés dans la figure 6.7.

Nous pouvons avancer que les pseudo-classes ne seront donc probablement pas très efficaces pour détecter les mots. Bien sûr, des techniques pourraient être utilisées pour améliorer ces scores, comme des modèles n-grams. Mais nous n'avons pas continué dans cette voie pour l'instant et effectué une dernière étude sur les mots dans la table 6.7.

La table 6.7 contrebalance un peu les résultats précédents. Celle-ci présente les ressemblances des pseudo-phones et des mots selon la taille des mots considérés

TABLE 6.6 – Écritures phonétiques du mot anglais « *something* » à gauche et indices des groupes correspondant aux segments phonétiques à droite

Classes des segments							Groupes attribués						
s	ah	m	n	ih	ng		24	7	3	9	13	13	
s	ah	m	iy				24	7	20	13			
s	ah	m	th	ih	ng		24	7	3	14	13	3	
s	ah	m	th	ih	ng		24	27	3	14	6	3	
s	ah	m	th	ih	ng		24	27	3	14	6	3	
s	ah	m	th	ih	n		24	7	3	9	13	3	
s	ah	m	th	ih	ng		19	7	3	9	6	20	
s	ah	m	ih	ng			24	7	11	13	3		
s	ah	m	th	ih			24	7	3	5	21		
s	ah	m	th	ih	ng		24	30	3	9	13	11	
s	ah	m	b	ih	ng		19	7	20	11	29	16	
s	ah	m	th	ih	ng		19	7	20	14	8	3	
s	ah	m	n	ihn			24	7	33	11	16		
s	ah	m	th	ih	ng		24	7	3	14	21	3	
s	ah	m	th	ih	ng	g	24	7	3	14	10	3	9
s	ah	m	ih	n			24	30	3	22	3		
s	ah	m	th	ih	n		1	2	3	26	10	3	
s	ah	m	b	ih	ng		24	23	20	9	10	3	
s	ah	m	th	ih	n		1	7	9	26	10	3	
s	ah	m	th	ih	ng		19	7	20	14	21	3	

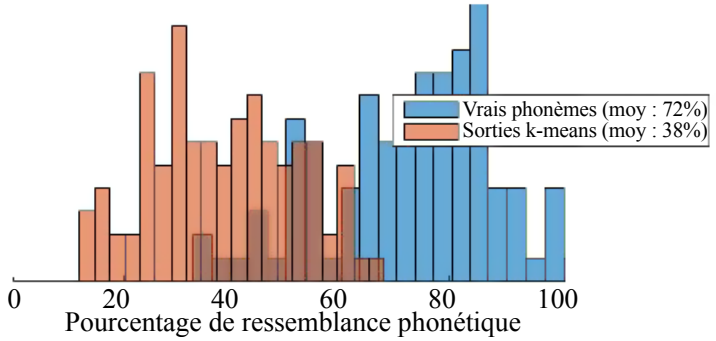

FIGURE 6.7 – Histogrammes des pourcentages de ressemblance phonétique des 100 mots les plus fréquents pour les vrais phonèmes (en bleu) et les pseudo-phones (en rouge)

et le nombre de mots égaux trouvés. Nous voyons que les mots les plus courts obtiennent les moins bons scores. Les phonèmes des mots de taille 3 sont la grande majorité des mots regardés dans cet exemple (>80% des exemples mots) et en moyenne moins d'un des trois phonèmes qui le composent est regroupé avec des exemples du même phonème et il n'y a que 7% de ressemblance entre les attributions des groupes d'exemples d'un même mot. Ces deux chiffres augmentent de plus 10% pour les mots de 4 phonèmes.

Si nous continuions dans cette voie et cherchions des pseudo-mots, nous devrions donc privilégier les plus longs, même s'ils sont beaucoup moins nombreux. Cependant, les pseudo-phones ne sont pas les seules données générées par notre

TABLE 6.7 – Taux de segments phonétiques (%) pour lesquels au moins 1 des n groupes les plus proches est majoritairement composé de segments de la même classe, avec n allant de 1 à 5 phonèmes

Mots		Ressemblance	
Taille	Nombre	phonèmes (%)	Mots (%)
7	6	79	79
6	28	77	78
5	182	59	43
4	1557	40	19
3	10240	29	7

modèle : il y a aussi de nouvelles représentations paramétriques, que nous avons évaluées sur les données du challenge ZRSC et dont nous allons donner les résultats dans la section suivante.

6.5 Évaluation des représentations paramétriques

Le but du challenge *Zero Resource Speech Challenge* (ZRSC) est d'apprendre des représentations capables de discriminer entre eux les différents phonèmes d'une langue.

Dans cette section, nous présentons d'abord les corpus du challenge avant de réaliser des expériences pour optimiser les hyperparamètres du système spécifiquement pour ce contexte. Puis, nous donnons les résultats obtenus avant de tenter une amélioration de notre système en utilisant un réseau de neurones multi-locuteurs.

6.5.1 Corpora du ZRSC 2017

Les participants ont reçu des fichiers audio bruts et des paramètres de bas niveau déjà extraits par les organisateurs du challenge, à savoir 13 MFCC et leurs premières et secondes dérivées.

À des fins de développement, trois grandes bases de données open-source ont été publiées en français, en anglais et en mandarin, ainsi que des transcriptions phonétiques manuelles ou alignées automatiquement au niveau du triphone.

Pour chaque langage, les données comprennent un ensemble d'entraînement et un ensemble de test découpé en trois sous-ensembles : fichiers de 1 seconde, de 10 secondes et de 120 secondes. Les locuteurs des ensembles de test sont différents de ceux des ensembles d'apprentissage. Les participants étaient libres d'utiliser les sous-ensembles d'entraînement ou non. De notre côté, nous n'avons utilisé que les sous-ensembles de test, car aucun gain de performance n'a été observé en considérant également les sous-ensembles d'apprentissage.

La table 6.8 affiche les statistiques de ces langues. Pour les trois langues, nous avons utilisé séparément 27 heures en anglais, 17 heures en français et 25 heures en mandarin. Il y a aussi deux langues de test, inconnues et notées « L1 » et « L2 », évaluées seulement par les organisateurs du challenge lors de la soumission.

Les résultats du challenge (DUNBAR et al. 2017) sont donnés par les organisateurs du challenge pour tous les ensembles de test : français, anglais, mandarin et les deux langues surprises, pour 1, 10 et 120 secondes [1]. Nous avons rapporté dans la figure 6.8 des extraits des résultats pour la *baseline*, la *topline*, les deux meilleurs

1. https://zerospeech.com/2017/results.html

TABLE 6.8 – Durées des corpus de test du Zero Resource Speech Challenge

Langage	Anglais	Français	Mandarin	L1	L2
Durée (heures)	27	17	25	11	6

participants et notre modèle initialement soumis. Les scores sont indiqués en taux d'erreur ABx.

Author / Affiliation	Development dataset									Surprise dataset						with supervision	rank
	English			French			Mandarin			LANG1			LANG2				
	1s	10s	120s	1s	10s	120s	1s	10s	120s	1s	10s	120s	1s	10s	120s		
Inter-locuteurs																	
Baseline MFCC	23.4	23.4	23.4	25.2	25.5	25.2	21.3	21.3	21.3	23.6	23.2	23.0	30.0	29.5	29.5	No	
Topline kaldi posteriors, HMM-GMM	8.6	6.9	6.7	10.6	9.1	8.9	12.0	5.7	5.1	12.8	10.5	10.4	7.1	3.6	4.3	Yes	
Pellegrini et al. - #2 IRIT Toulouse	17.6	16.2	16.3	20.1	17.7	17.3	14.7	13.5	13.4	19.2	16.3	16.0	23.3	23.3	23.1	No	10
Heck et al. NAIST	10.1	8.7	8.5	13.6	11.7	11.3	8.8	7.4	7.3	11.9	10.0	9.7	13.0	10.0	9.9	No	1
Chen et al. - #2 Northwestern Polytechnical University	12.7	11.0	10.8	17.0	14.5	14.1	11.9	10.3	10.1	14.7	11.7	11.6	16.9	14.7	14.4	No	2
Intra-locuteurs																	
Baseline MFCC	12.0	12.1	12.1	12.5	12.6	12.6	11.5	11.5	11.5	10.3	9.3	9.4	14.1	14.3	14.1	No	
Topline kaldi posteriors, HMM-GMM	6.5	5.3	5.1	8.0	6.8	6.8	9.5	4.2	4.0	8.7	7.1	7.0	6.6	4.6	3.4	Yes	
Pellegrini et al. - #2 IRIT Toulouse	9.8	8.1	8.2	11.6	9.5	9.3	10.9	8.4	8.1	8.8	6.6	6.3	13.1	11.7	11.7	No	10
Heck et al. NAIST	6.9	6.2	6.0	9.7	8.7	8.4	8.8	7.9	7.8	6.5	5.6	5.3	10.9	8.8	8.4	No	1
Chen et al. - #2 Northwestern Polytechnical University	8.5	7.3	7.2	11.2	9.4	9.4	10.5	8.7	8.5	7.6	6.2	6.1	11.6	9.8	9.6	No	2

FIGURE 6.8 – Table extrait des résultats du ZRSC 2017 (taux d'erreur ABx) : la *baseline*, la *topline*, les deux meilleurs participants et notre modèle initial soumis pour l'intra-locuteur (*within*) et l'inter-locuteurs (*across*)

La *baseline* correspond à des MFCC et leurs deux premières dérivées. La *topline* correspond aux posteriorgrams phonétiques, obtenus par un décodeur phonétique supervisé entraîné sur des données annotées manuellement via l'outil kaldi (POVEY, GHOSHAL et al. 2011). Les MFCC permettent à eux seuls d'obtenir pour les trois langues aux environs de 12% de taux d'erreur ABx intra-locuteur (*within*) et 20 à 25% de taux d'erreur ABx inter-locuteurs (*across*). Le modèle supervisé obtient moins de 7% d'erreur pour le *within* et moins de 10% d'erreur pour le *across*.

Nous constatons un écart maximal entre les différentes langues d'environ 5% pour l'intra-locuteur (*within*) et 6% pour l'inter-locuteurs (*across*). Les langues dont les scores sont les plus éloignées semblent être les deux langues surprises, qui sont toutes deux à l'opposé. Les langues d'entraînement se situent dans une fourchette plus petite. La langue 2 est celle qui a le plus grand écart entre la *baseline* (MFCC), pour laquelle elle obtient de loin les moins bons résultats, et la *topline* (posteriorgrammes supervisés), pour laquelle elle a des scores environs deux fois inférieurs

aux autres langues. Les résultats diffèrent aussi selon les tailles de fichiers : les sous-ensembles contenant les fichiers de seulement 1 seconde ont les moins bons scores, même avec les MFCC.

Les résultats des principaux systèmes soumis sont décrits plus amplement dans la section 6.5.3.

Dans le reste de ce document, nous nous concentrons uniquement sur les ensembles de test de 10 secondes (taille médiane) pour les langues d'entraînement (français, anglais et mandarin).

6.5.2 Optimisation du système

Nous avons remarqué que le taux d'erreur ABx obtenu par les paramètres dépend beaucoup des traitements appliqués aux paramètres. Un blanchiment à l'aide d'une ZCA a amélioré leur score de 2% pour le score intra-locuteur et de 3% pour le score inter-locuteurs. Le calcul d'une ZCA nécessite un paramètre (epsilon) dont la valeur a beaucoup influencé nos résultats, comme nous pouvons le voir sur la figure 6.9. Quelle que soit sa valeur (comprise entre 0 et 1), les paramètres blanchis ont obtenus de meilleurs résultats pour le challenge mais les résultats du k-means appliqué sur ces paramètres se sont avérés davantage dépendants d'un bon choix d'epsilon. La figure 6.9 montre aussi qu'il n'existe pas d'epsilon idéal pour un même ensemble de données : les scores intra-locuteur et inter-locuteurs ne sont pas minimaux pour les mêmes valeurs d'epsilon. La marge de manœuvre est très faible : le score pour l'intra-locuteur notamment descendant puis remontant très rapidement avec l'augmentation des valeurs d'epsilon.

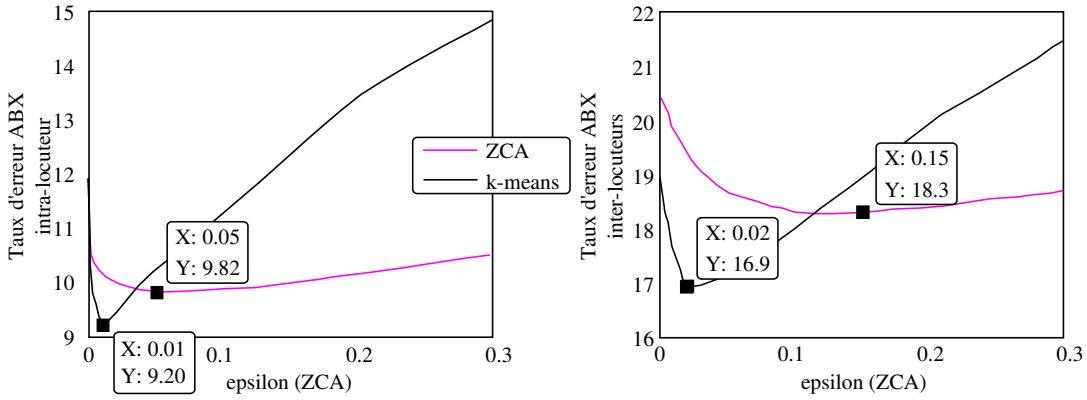

FIGURE 6.9 – Évolution du taux d'erreur ABx des paramètres blanchis par ZCA (en mauve) puis des paramètres générés par un k-means (en noir) en fonction de la valeur de l'hyperparamètre epsilon de la ZCA, avec les mesures intra-locuteur (*à gauche*) et les mesures inter-locuteurs (*à droite*), pour le corpus de test en français

Durant nos expériences nous avons pu constater que l'epsilon idéal variait aussi selon les langues. Pour essayer de trouver le meilleur epsilon possible quelles que soient les conditions, nous avons regardé dans la figure 6.10 les valeurs prises par les valeurs propres auxquelles epsilon est ajouté. Le graphique de gauche est l'histogramme d'un des fichiers audio (d'un locuteur) de 10 secondes, représentatif de la majorité des fichiers pour cette langue et ces paramètres. Deux lots se distinguent, un lot de valeurs propres élevées à conserver et un lot de valeurs propres faibles à écraser. La valeur idéale pour l'epsilon se situera plus ou moins entre ces deux sommets, peut-être un peu plus vers les valeurs élevées puisque c'est un seuillage

doux. Pour vérifier que tous les locuteurs de ce corpus donnaient des histogrammes similaires nous avons affiché l'histogramme de toutes les valeurs propres des différents fichiers dans le graphique de droite. Les deux concentrations de valeurs se démarquent toujours, avec quelques exceptions autour qui nous ont fait choisir une valeur différente d'epsilon par fichier plutôt qu'une valeur globale qui ne conviendrait pas à des exceptions.

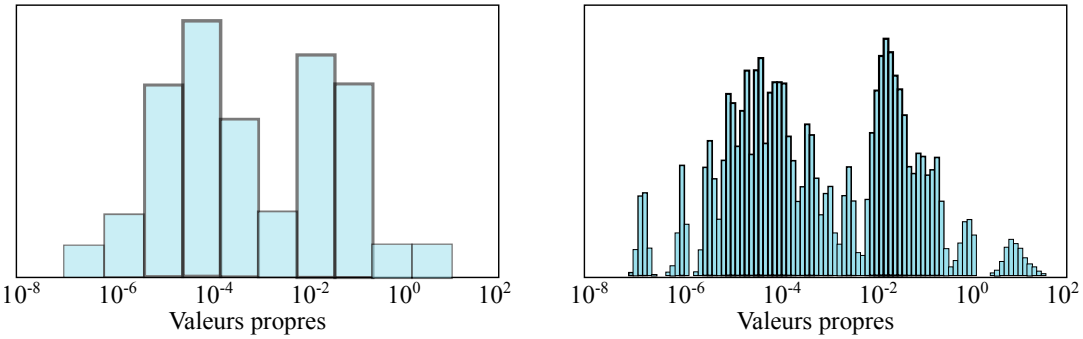

FIGURE 6.10 – Répartition des valeurs propres de la matrice de corrélation d'un locuteur (à gauche) et de celles de toutes les données (à droite) pour le corpus de test en français

Pour trouver un calcul d'epsilon non supervisé permettant d'obtenir pour chaque fichier une valeur proche d'une valeur optimale, nous avons fait plusieurs expériences dont les résultats sont donnés dans la table 6.9. Le taux d'erreur ABx a jusqu'à 4% de variation selon la valeur d'epsilon choisie. L'impact est généralement inverse pour les conditions intra-locuteur et inter-locuteurs, et de même différent entre les paramètres blanchis et ceux du k-means. Nous avons choisi d'utiliser le troisième quartile par la suite grâce à ses bons résultats avec le k-means, dont les pseudo-labels générés sont utilisés ensuite par le réseau de neurones.

TABLE 6.9 – Taux d'erreur ABx intra- et inter-locuteurs pour 2 des langues de développement du ZRSC2017. Les paramètres sont blanchis par ZCA puis projetés dans l'espace créé par les centroïdes d'un k-means appliqué dessus. La valeur du paramètre epsilon de la ZCA est recalculée pour chaque fichier (avec 1 locuteur/fichier)

Calcul d'epsilon selon les valeurs propres	Mandarin		Anglais	
	ZCA	k-means	ZCA	k-means
moyenne	9,3 - **13,7**	12,9 - 15,2	9,2 - **16,2**	11,4 - 17,5
médiane	**8,1** - 14,9	**8,6** - 13,7	9,2 - 19,6	9,2 - 17,4
troisième quartile	**8,1** - 14,4	9,1 - 13,3	**8,7** - 17,7	8,7 - **15,2**
neuvième décile	**8,1** - 14,5	9,1 - **13,1**	8,8 - 17,8	**8,4** - 15,5

Nous avons aussi testé des paramètres issus d'auto-encodeurs. Nous avons réalisés différents tests, avec différents réseaux, en faisant notamment varier la fonction d'activation de la couche générant les paramètres. Malheureusement, plus nous améliorons les paramètres pour le challenge, plus nous diminuons la qualité des paramètres issus d'un k-means appris dessus. La table 6.10 montre un exemple de résultats pour un Auto-Encodeur générant des paramètres meilleurs que les paramètres de base selon le critère du taux d'erreur ABx. Les paramètres du k-means ont détérioré de 20% les résultats et même si leur blanchiment par ZCA a été efficace,

les paramètres de l'Auto-encodeur sont meilleurs bruts et ne sont pas améliorés par notre système.

TABLE 6.10 – Taux d'erreur ABx intra- et inter-locuteurs pour les ensembles de test des langues de développement du ZRSC2017. Les paramètres sont issus d'un AE puis modifiés : blanchis puis projetés dans l'espace créé par les centroïdes d'un k-means

	Français	Anglais	Mandarin
AE	**9,9 - 17,1**	**8,2 - 15,9**	**8,4 - 12,8**
+k-means	29 - 39	27 - 37	22 - 32
+ZCA	11,9 - 19	9,8 - 16,1	10,5 - 13,7

6.5.3 Résultats

Les taux d'erreur ABx obtenus pour les 3 langues à chaque itération sont affichés dans la figure 6.11. Nous voyons que dans les deux cas (intra- et inter-locuteurs) et pour les trois langues (anglais, français, mandarin) la première itération (k-means sur les posteriorgrammes du CNN) est efficace. L'intérêt des autres itérations dans le cas du intra-locuteur est ensuite moindre, contrairement au inter-locuteurs qui continue de diminuer après la première itération.

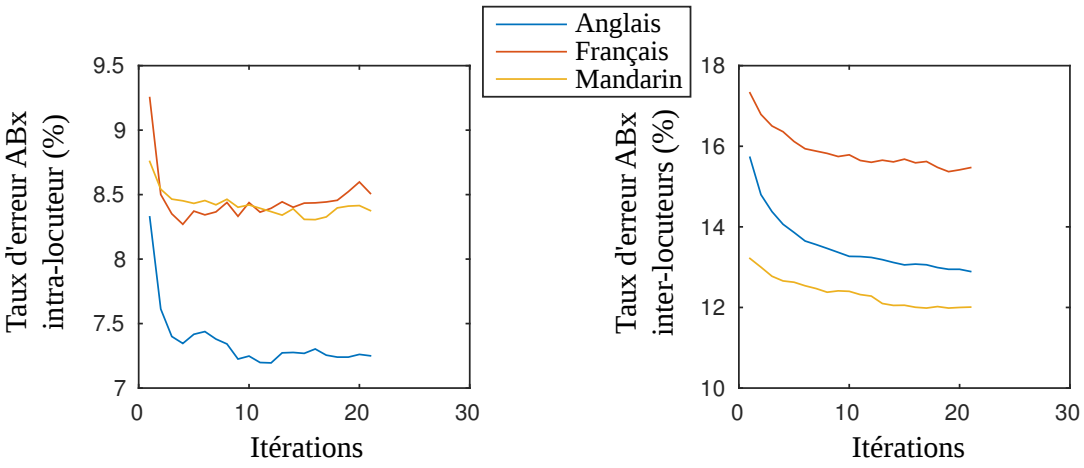

FIGURE 6.11 – Évolution du taux d'erreur ABx au cours des itérations

Les distances des centroïdes des k-means entre les paramètres issus d'un CNN génèrent donc des paramètres efficaces au sens du taux d'erreur ABx. Mais les corrections des classes du k-means apportées n'améliorent pas les résultats des paramètres du nouveau CNN pour la condition intra-locuteur. Par contre, ces paramètres deviennent davantage robustes au locuteur à chaque itération.

Les différents participants du challenge ont obtenu des résultats compris entre la *baseline* et la *topline*. Le participant arrivé premier (HECK, SAKTI et NAKAMURA 2017) se démarque particulièrement pour ses résultats très robustes au locuteur en étant seulement 2% derrière la *topline* pour le *across* comme pour le *within*. Le participant arrivé deuxième est seulement 1% derrière lui en *within* mais 3% moins bon en *across*. Notre système se situe au niveau du deuxième pour le *within* mais est moins bon pour le *across*, avec 2% d'erreur de plus. C'est donc en *across* que notre modèle est moins bon, probablement parce que nous n'utilisons pas d'étapes d'adaptation

TABLE 6.11 – Résultats du *Zero Resource Speech Challenge* en terme de taux d'erreur ABx

	Intra-locuteur (*Within*)					Inter-locuteurs (*Across*)				
	En	Fr	Ma	L1	L2	En	Fr	Ma	L1	L2
Baseline [2]	12,1	12,6	11,5	9,3	14,3	23,4	25,5	21,3	23,2	29,5
Topline [3]	5,3	6,8	4,2	7,1	4,6	6,9	9,1	5,7	10,5	3,6
Premier [4]	6,2	8,7	7,9	5,6	8,8	8,7	11,7	7,4	10,0	10,0
Deuxième [5]	7,3	9,4	8,7	6,2	9,8	11,0	14,7	10,3	11,7	14,7
IRIT [6]	8,2	9,7	8,5	6,6	11,7	16,3	17,6	13,5	16,3	23,3
Notre système	7,3	8,5	8,4	-	-	12,9	15,5	12,0	-	-

au locuteur contrairement aux meilleurs concurrents. Pour information, notre soumission (IRIT) est également indiquée. Ses résultats sont meilleurs que la *baseline* mais moins bon que les meilleurs concurrents et que notre système actuel.

Pour approfondir ce point, nous avons essayé d'appliquer des méthodes d'adaptation. Nous avons testé la VTLN (WEGMANN et al. 1996), comme l'ont fait les deux participants arrivés premiers. Nous avons également testé la CMVN (HAEB-UMBACH 1999) car il s'agit d'une technique de normalisation très répandue dans le traitement de la parole. Les résultats obtenus sur le sous-ensemble de test français (fichiers de 10s) sont présentés dans le tableau 6.12.

TABLE 6.12 – Comparaison entre différentes techniques de normalisation (selon le taux d'erreur ABx, sur l'ensemble de test en français d'extraits de 10s), avec (1) : les paramètres acoustiques ; (2) : les paramètres des k-means pour 100 regroupements

	MFCCs		ZCA		VTLN		CMVN	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
Intra-locuteur	12,1	26,1	10,2	9,7	11,9	12,4	11,6	15,0
Inter-locuteurs	26,6	35,9	19,8	17,6	24,0	22,9	23,9	26,5

Comme nous pouvons le constater, les MFCC bruts donnent de très mauvais résultats avec des k-means. Au contraire, le VTLN et la ZCA se comportent bien avec k-means. Les paramètres avec VTLN ont certes apporté un léger gain par rapport aux MFCC bruts, mais sont toujours moins bons que la ZCA, avec et sans k-means. Il est intéressant de noter que le k-means dégrade légèrement l'ABx intra-locuteur puisque nous perdons certaines informations spécifiques au locuteur en appliquant le VTLN, alors que l'ABx inter-locuteurs est amélioré grâce à des regroupements plus indépendants du locuteur. L'application de la ZCA sur les MFCC-VTLN (non reportés dans le tableau) améliore encore les résultats : intra-locuteur : 10,7%, inter-locuteurs : 19,2%.

Dans cette comparaison, l'utilisation d'un k-means sur les ZCA-MFCC donne toujours la meilleure représentation.

2. (DUNBAR et al. 2017)
3. (DUNBAR et al. 2017)
4. (HECK, SAKTI et NAKAMURA 2017)
5. (H. CHEN et al. 2017)
6. (PELLEGRINI, MANENTI et PINQUIER 2017)

6.5.4 Ouverture : réseau multi-locuteurs

Le point faible de notre système étant les résultats inter-locuteurs, nous avons travaillé dessus. Nous avons d'abord regardé si le problème pouvait venir d'une répartition inégale des locuteurs dans les regroupements mais les répartitions se sont avérées uniformes sauf quelques groupes dont la suppression n'a pas amélioré les résultats.

Les réseaux de neurones ayant la réputation d'une très bonne portabilité, c'est donc notre réseau que nous avons cherché à améliorer. En nous inspirant de la littérature nous avons opté pour un réseau multi-tâches en considérant la prédiction des groupes de chaque locuteur comme une tâche différente. La séparation des locuteurs se fait uniquement sur la couche de sortie, comme illustré dans le graphique 6.12. Les paramètres générés pour calculer le taux d'erreur ABx sont, comme précédemment, issus de la dernière couche cachée.

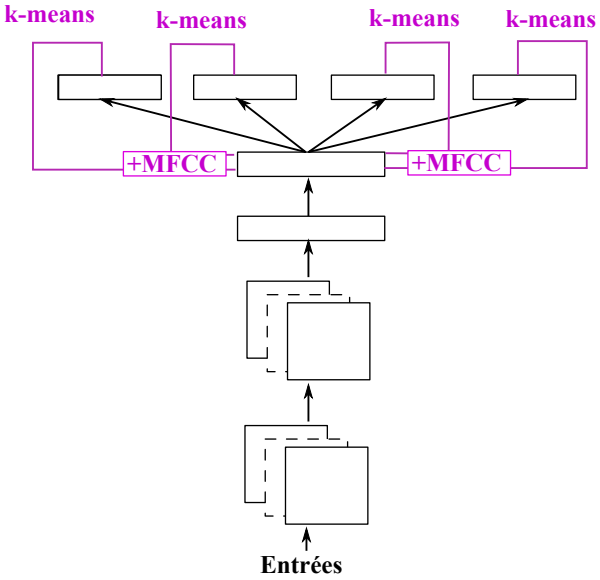

FIGURE 6.12 – Illustration de la structure du CNN multi-tâches apprenant sur les classes générées avec un k-means différent par locuteur

Malgré les divers ajustements du réseau testé, nous n'avons pas réussi à obtenir des résultats satisfaisants. Des exemples d'évolution du taux d'erreur ABx au cours des itérations sont affichés dans la figure 6.13.

Nous avons utilisé le réseau précédemment appris (dont l'évolution au cours des itérations est affiché en noir, avec des indices d'itérations négatifs). Les poids de ce réseau servent d'initialisation aux poids des réseaux multi-locuteurs testés, en-dehors des dernières couches parallèles spécifiques à ce réseau. Nous lançons ensuite l'apprentissage du réseau multi-locuteurs et obtenons des évolutions du taux d'erreur ABx différentes selon les hyperparamètres choisis, dont certaines sont affichées dans la figure 6.13 (courbes de couleur, avec indices d'itérations positifs). Nous voyons que les résultats intra-locuteur ont toujours été pénalisés par le système multi-locuteurs. Les résultats inter-locuteurs ont moins été pénalisés et ont même au mieux été à peu près constants, quand nous n'avons réappris que la dernière couche cachée. Cependant, les résultats obtenus par notre système précédent n'ont pas étaient améliorés.

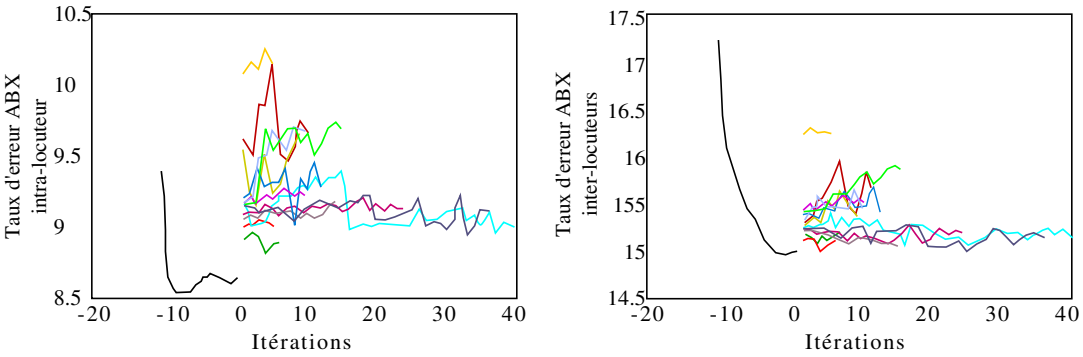

FIGURE 6.13 – Résultats d'évolution du taux d'erreur ABx en fonction des itérations pour différents hyperparamètres du réseau pré-entraîné par l'apprentissage non multi-tâches précédent

6.6 Conclusion

Dans ce chapitre, nous décrivons nos expériences de découverte d'unités de parole et d'apprentissage de représentation fondées sur des CNN entraînés sur des pseudo-phones obtenus par k-means, dans une boucle itérative alternant regroupement et classification. Tout d'abord, nous avons défini la configuration du système (nombre de couches, tailles de filtres, etc.). Nous avons ajusté les hyperparamètres en utilisant un corpus de parole en français (BREF80) et une métrique de pureté des regroupements au niveau du phonème. Nous rapportons des résultats de pureté sur le français (BREF80), l'anglais américain (BUCKEYE) et une langue moins représentée, le xitsonga (NCHLT). Ceux-ci sont supérieures à 40% pour le français et le xitsonga et d'environ 30% pour l'anglais. Cet écart peut s'expliquer par la différence du type de parole : parole lue pour BREF80 et NCHLT et parole conversationnelle dans le cas de BUCKEYE.

Ensuite, nous avons évalué notre approche sur les données du *Zero Resource Speech Challenge* (anglais, français et mandarin) pour la tâche consistant à générer de manière non supervisée de nouvelles représentations qui sont phonétiquement pertinentes en fonction du taux d'erreur ABx. Les probabilités extraites directement des GMM ou des CNN, appelées posteriorgrammes, ne se sont pas révélées efficaces dans notre cas. Pour le challenge ZRSC 2017, nous avons soumis des représentations fondées uniquement sur k-means : il s'agissait des distances entre les vecteurs MFCC blanchis par ZCA et les centroïdes des groupes. Le regroupement en alternance avec k-means et la classification avec un CNN ont surpassé la représentation obtenue par la seule application d'un k-means.

Ces représentations ont donné des résultats de taux d'erreur ABx meilleurs que la soumission faite par le participant classé deuxième pour la condition de discrimination intra-locuteur : pour le français et le mandarin, respectivement les valeurs de taux d'erreur ABx de 8,5 et 8,4 % ont été obtenues par rapport à 9,4 et 8,7%. Nos résultats sont légèrement moins bons dans la condition inter-locuteurs, dans laquelle le taux d'erreur est estimé entre les locuteurs. Notre système résiste donc moins à la variabilité des locuteurs que les systèmes arrivés premiers. Former des couches de sortie dépendantes du locuteur dans un réseau CNN, comme cela se fait dans un réseau neuronal multilingue, ne nous a pas apporté davantage d'indépendance au locuteur. Un travail futur pourrait viser à améliorer les capacités de notre approche indépendante du locuteur, en testant des méthodes de normalisation et d'adaptation au locuteur propres aux réseaux de neurones, comme l'on peut en trouver dans la

littérature récente (YOSHIOKA, RAGNI et GALES 2014 ; PARTHASARATHI et al. 2015 ; LIAO 2013 ; SWIETOJANSKI et RENALS 2014). Ainsi, des techniques de régression linéaire du maximum de vraisemblance (Maximum Likelihood Linear Regression, MLLR) (FERRAS et al. 2007) peuvent être utilisées sur les paramètres d'entrée du réseau de neurones (YOSHIOKA, RAGNI et GALES 2014 ; PARTHASARATHI et al. 2015). Nous pouvons citer également des normalisations lors de l'apprentissage du réseau en ajoutant un terme de pénalité à la fonction de coût (LIAO 2013) ou encore l'estimation d'un paramètre supplémentaire dépendant du locuteur pour chaque neurone (SWIETOJANSKI et RENALS 2014).

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

Le but de cette thèse était la découverte d'unités et de sous-unités linguistiques. Nous nous sommes principalement focalisés sur l'utilisation des réseaux de neurones pour la découverte non supervisée de sous-unités linguistiques et de paramètres discriminants pour cette tâche.

Nous avons abordé le sujet de cette thèse par une approche supervisée de reconnaissance des phonèmes, nous permettant d'ajuster les hyperparamètres de nos réseau de neurones. Pour nous aider à optimiser nos résultats, comme certains travaux l'ont fait, nous avons développé une méthode de segmentation en phonèmes faiblement supervisée et inter-langues (portable d'une langue à une autre). Puis nous avons abordé la partie non supervisée de cette thèse, en testant des paramètres que les réseaux de neurones non supervisés pouvaient nous apporter puis en créant une architecture mêlant une méthode de regroupement et une méthode de classification.

7.1.1 Classification phonétique supervisée

L'étude de la reconnaissance phonétique supervisée par réseau de neurones avait pour but d'appréhender les réseaux de neurones et de développer un réseau simple pouvant nous servir d'outil pour la suite de la thèse, c'est-à-dire pour l'apprentissage non supervisé. Nous nous sommes tournés vers les réseaux denses puis les réseaux convolutionnels, utilisant des couches de filtres en entrée.

Sur l'ensemble BUCKEYE, un corpus de parole conversationnelle en anglais, nous avons obtenu environ 60% de taux de classification. La structure de réseau construite nous a ensuite été utile dans notre modèle de découverte non supervisée de pseudo-phones et de génération non supervisée de nouvelles représentations paramétriques. Nous avons vu dans certains articles étudiés dans l'état de l'art, comme (ONDEL, Lukáš BURGET et ČERNOCK 2016), que la segmentation phonétique peut être utile à des modèles non supervisés de découverte de sous-unités lexicales. Nous avons donc construit un réseau réalisant cette tâche.

7.1.2 Segmentation phonétique supervisée

Pour réaliser une segmentation automatique, nous avons construit un réseau de neurones convolutionnel entraîné à prédire les frontières entre les phonèmes (MANENTI, PELLEGRINI et PINQUIER 2016b ; MANENTI, PELLEGRINI et PINQUIER 2016a). Une des difficultés était la répartition inégale des éléments des deux classes apprises (frontière/non frontière), la nécessité de précision et d'unicité des prédictions des frontières. Pour qu'une seule trame soit classée « frontière », nous avons proposé un

post-traitement utilisant une convolution avec une fenêtre de Hamming et une détection des sommets. Pour remédier aux inégalités de nombre d'éléments entre les deux classes, une méthode peut être de donner davantage d'importance aux éléments de la classe la moins présente. Dans notre travail, nous avons plutôt choisi d'attribuer la classe « frontière » aux trames voisines, permettant ainsi de prendre en considération le recouvrement entre les fenêtres. Notre système de post-traitement précédemment décrit, utilisant le moyennage de la courbe des probabilités de prédiction et de détection de sommets, a permis de réduire les attributions de classes « frontière » à une seule trame. Utiliser des fenêtres de 16 ms avec un pas de 4 ms a rendu notre réseau plus précis et nous a permis d'améliorer nos résultats.

Cette approche obtient d'excellents résultats en segmentation phonétique : 79% (resp. 68%) de F-mesure pour 20 ms (resp. 10 ms) de marge sur le corpus de parole conversationnelle en anglais BUCKEYE. En comparaison, notre segmentation est de qualité similaire à la segmentation manuelle selon l'accord inter-annotateurs fourni par les créateurs du corpus pour une marge d'erreur de 20 ms et meilleur pour une marge de 10 ms. Notre système est donc plus précis qu'un annotateur humain lorsqu'il place correctement une frontière. De plus, il n'a pas besoin de beaucoup de données (environ une dizaine de minutes par locuteur et 5 locuteurs différents) pour être performant. Nos expériences ont montré qu'il est portable à d'autres langues, notamment pour des langues peu dotées telle que le xitsonga.

Nous avons évalué l'utilité de notre segmentation automatique dans le chapitre 6 dans le modèle de découverte de pseudo-phones. Son utilisation permet d'améliorer nos résultats de quelques pourcentages. Cependant, la segmentation n'étant pas non supervisée, elle ne rentre pas dans les critères d'un système non supervisé. Nous ne l'avons donc pas utilisée dans nos modèles totalement non supervisés.

7.1.3 Réseaux de neurones non supervisés

Les réseaux de neurones non supervisés n'ont pas l'information des classes phonétiques pour s'entraîner. Ils doivent se restreindre à utiliser les seules informations fournies, c'est-à-dire les paramètres d'entrée. Les Auto-Encodeurs sont des réseaux non supervisés qui apprennent à reconstruire les trames d'entrées, après plusieurs projections non linéaires effectuées dans leurs couches cachées. Nous avons utilisés ces réseaux pour générer de nouveaux paramètres à partir des MFCC. Le réseau s'est avéré efficace pour compresser l'information issue de la trame d'entrée et des trames voisines et nous avons obtenu des paramètres plus efficaces que les MFCC pris en entrée d'un outil de classification phonétique : 45,7% contre 33,9% pour un même nombre de paramètres. Cependant, nous avons rencontré des difficultés pour l'apprentissage de ces AE. En effet, toutes les règles de mise à jour des poids ne conviennent pas, comme par exemple le nesterov momentum, alors qu'il nous avait permis d'obtenir des résultats similaires aux autres règles de mise à jour dans les cas supervisés testés (classification et segmentation). De même, les bancs de filtres se sont avérés ne pas être une entrée adaptée pour réaliser ce travail d'extraction non supervisée d'information, contrairement aux cas supervisés testés.

D'autres expériences ont été réalisées avec les réseaux de neurones non supervisés. Ainsi, nous avons obtenu une bonne compression du signal brut de la parole par AE, obtenant un faible taux d'erreur de reconstruction (RMS), même avec une compression de 99%. Mais les paramètres générés par notre réseau ne correspondent pas à des paramètres acoustiques pertinents et ne semblent pas avoir d'autre utilité que la compression.

Fondée sur une approche différente, une pré-étude nous a permis d'obtenir des pseudo-poids LPC. Il s'agit de poids de neurones d'un réseau fonctionnant comme un LPC, c'est-à-dire apprenant à prédire la valeur suivante du signal audio à l'aide des valeurs précédentes. L'avantage le plus immédiat de ces poids est qu'ils sont plus rapides à obtenir que les vrais paramètres LPC, dont la complexité est en $O(p^3)$, avec p le nombre de paramètres calculés. Au contraire, la complexité des pseudo-LPC estimés par réseaux de neurones est linéaire. Ces paramètres pourraient donc être utiles dans des cas où beaucoup de coefficients sont calculés, comme par exemple dans certains travaux sur la réverbération (KINOSHITA et al. 2009).

7.1.4 Découverte de pseudo-phones et génération non supervisée de nouvelles représentations paramétriques

Pour finir, nous avons proposé un système de découverte non supervisée de sous-unités linguistiques et de représentations paramétriques en utilisant alternativement une méthode de regroupement (k-means) et un réseau de neurones (convolutionnel, avec les couches cachées prenant en entrée les sorties de plusieurs des couches précédentes, de type similaire à un denseNet) (MANENTI, PELLEGRINI et PINQUIER 2017 ; MANENTI, PELLEGRINI et PINQUIER p.d.).

Le k-means découvre des groupes, qui ont une pureté d'environ 30% sur le corpus conversationnel testé (BUCKEYE en anglais) et entre 40% et 50% sur les corpus de parole lue testés (BREF en français, NCHLT en xitsonga). Ces groupes servent à entraîner le réseau de neurones qui fournit ensuite de meilleurs paramètres, générés par ses couches cachées. Ceux-ci servent à leur tour à améliorer la découverte de groupes par le k-means, qui obtient alors de meilleurs résultats. Ce cercle vertueux permet d'obtenir des paramètres discriminants pour des phonèmes d'un même locuteurs sur les données du *Zero Ressource Speech Challenge* (ZRSC) : de faibles taux d'erreur ABx intra-locuteur, compétitifs par rapport aux meilleurs participants du challenge.

7.2 Perspectives

Notre travail peut évoluer et être amélioré de différentes façons. Voici quelques pistes possibles pour des travaux futurs.

Nous avons constaté que notre modèle de génération de nouvelles représentations paramétriques rencontrait des difficultés à être robuste au locuteur malgré les expériences avec des méthodes d'adaptation habituelles et utilisées par d'autres modèles soumis au challenge ZRSC. Il existe dans la littérature des méthodes d'adaptation propres aux réseaux de neurones (YOSHIOKA, RAGNI et GALES 2014 ; PARTHASARATHI et al. 2015 ; LIAO 2013 ; SWIETOJANSKI et RENALS 2014). Des techniques de régression linéaire du maximum de vraisemblance (Maximum Likelihood Linear Regression, MLLR) (FERRAS et al. 2007) peuvent être utilisées sur les paramètres d'entrée du réseau de neurones (YOSHIOKA, RAGNI et GALES 2014 ; PARTHASARATHI et al. 2015). Nous pouvons citer également des normalisations lors de l'apprentissage du réseau en ajoutant un terme de pénalité à la fonction de coût (LIAO 2013) ou encore l'estimation d'un paramètre supplémentaire dépendant du locuteur pour chaque neurone (SWIETOJANSKI et RENALS 2014). Une perspective d'amélioration de notre modèle est d'expérimenter de telles méthodes, peut-être d'abord dans un cadre supervisé comme nous l'avons fait au début de cette thèse pour la construction du réseau de neurones.

Un autre élément qui pourrait être ajouté à notre modèle non supervisé est un modèle au-dessus du modèle acoustique au niveau phonétique. Nous avons, en effet, pu constater l'efficacité de l'alternance de l'étape de regroupement et de classification dans notre modèle. La littérature va également dans ce sens, à l'aide d'échelles différentes dans certains cas. En particulier, le vainqueur du challenge ZRSC 2017 a utilisé plusieurs niveaux : monophones et triphones pour le modèle acoustique, puis un modèle de langage (HECK, SAKTI et NAKAMURA 2017), comme illustré sur la figure 7.1 dans le cadre vert. Nous pourrions donc ajouter une étape supplémentaire dans notre boucle itérative pour accentuer encore davantage son effet bénéfique sur les résultats. Ceci permettrait d'alterner la génération de nouveaux paramètres, la découverte de sous-unités lexicales et d'unités lexicales.

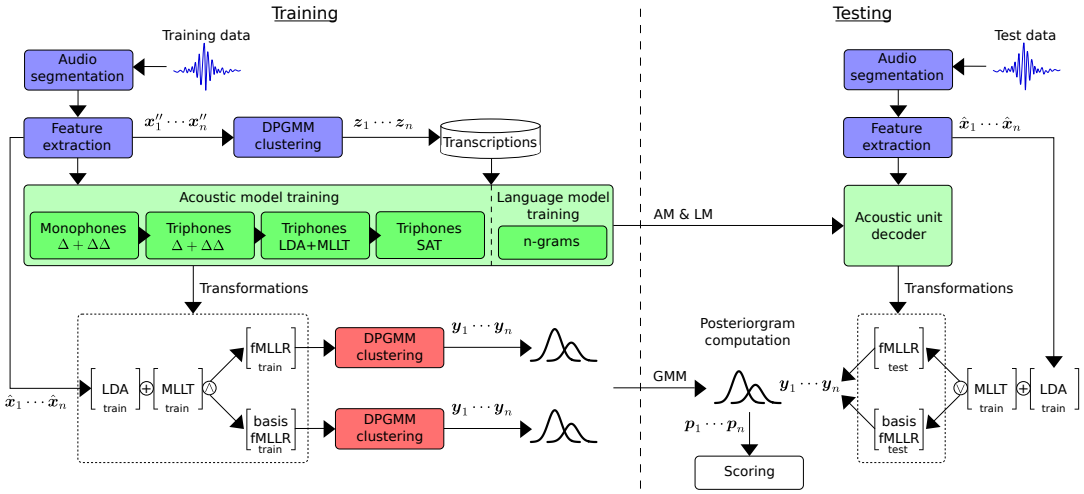

FIGURE 7.1 – Image issue de (HECK, SAKTI et NAKAMURA 2017) illustrant le système de génération non supervisée de paramètres ayant obtenu les meilleurs résultats au ZRSC 2017

Outre le fait d'avoir des paramètres et des modèles plus robustes au locuteur et une étape supplémentaire dans la boucle itérative, nous pourrions aussi travailler à utiliser de nouveaux paramètres, comme ceux générés par des AE ou ceux cherchant à imiter les LPC. Nous pourrions optimiser la structure de notre AE avec la métrique du challenge (le taux d'erreur ABx) et conjointement à notre modèle itératif pour que les paramètres générés soient davantage adaptés à ce problème. Continuer notre pré-étude sur les paramètres générés par réseaux de neurones imitant les paramètres LPC demandera davantage de temps, le travail étant moins avancé. Des pistes de recherche peuvent être étudiées, comme essayer de maintenir les racines du polynôme formé par les poids du réseau dans le cercle unité grâce à l'ajout de contraintes lors de l'apprentissage sur la valeur des poids ou l'ajustement du coût pénalisant davantage les valeur prédites supérieures (en absolu) que celles inférieures.

Le challenge ZRSC est pour nous un bon indicateur de l'état de l'art dans le domaine. Notre soumission à la version 2017 nous a également servi de « *baseline* » pour cette thèse. Ainsi, toutes nos nouvelles propositions et/ou modifications effectuées, pourrons être évaluées via les données du challenge ZRSC. Nous pourrons ainsi comparer les résultats avec notre modèle actuel pour vérifier l'efficacité des améliorations apportées.

En plus d'optimiser notre système actuel avec les pistes précédentes, nous pourrions l'augmenter (synthèse vocale, portabilité entre locuteurs) pour répondre à la

tâche de la version 2019 du challenge [1]. En effet, cette année la tâche inclut de la synthèse vocale : le but est la création d'un modèle TTS (Text To Speech), mais sans le T (texte) !

1. https://zerospeech.com/2019/

Bibliographie

ABADI, Martın et al. (2016). « Tensorflow : a system for large-scale machine learning ». In : *Proc. OSDI*. T. 16, p. 265–283.

ADDA, Gilles, Martine ADDA-DECKER et al. (2016). « Innovative technologies for under-resourced language documentation : The BULB Project ». In : *Proc. LREC, Workshop CCURL*, p. 59–66.

ADDA, Gilles, Sebastian STÜKER et al. (2016). « Breaking the unwritten language barrier : The BULB project ». In : *Procedia Computer Science* 81, p. 8–14.

AGENBAG, Wiehan et Thomas NIESLER (2015). « Automatic segmentation and clustering of speech using sparse coding and metaheuristic search ». In : *Proc. INTERSPEECH*. IEEE, p. 3184–3188.

AMIRIPARIAN, Shahin et al. (2017). « Sequence to sequence autoencoders for unsupervised representation learning from audio ». In : *Proc. DCASE*, p. 17–21.

ANDÉN, Joakim et Stéphane MALLAT (2014). « Deep scattering spectrum ». In : *Transactions on Signal Processing* 62.16, p. 4114–4128.

ANDRE-OBRECHT, Regine (1988). « A new statistical approach for the automatic segmentation of continuous speech signals ». In : *Transactions on Acoustics, Speech, and Signal Processing* 36.1, p. 29–40.

ANSARI, TK et al. (2017). « Unsupervised HMM posteriograms for language independent acoustic modeling in zero resource conditions ». In : *Proc. ASRU*. IEEE, p. 762–768.

ANTONIAK, Charles E (1974). « Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems ». In : *The annals of statistics*, p. 1152–1174.

ARISOY, Ebru et Murat SARAÇLAR (2015). « Multi-stream long short-term memory neural network language model ». In : *Proc. INTERSPEECH*. IEEE, p. 1413–1417.

ASSAYAG, Michel et al. (2015). « Meeting Assistant Application ». In : *Proc. INTERSPEECH*. IEEE, p. 734–735.

BADINO, Leonardo (2016). « Phonetic Context Embeddings for DNN-HMM Phone Recognition ». In : *Proc. INTERSPEECH*. IEEE, p. 405–409.

BADINO, Leonardo, Claudia CANEVARI et al. (2014). « An auto-encoder based approach to unsupervised learning of subword units ». In : *Proc. ICASSP*. IEEE, p. 7634–7638.

BADINO, Leonardo, Alessio MERETA et Lorenzo ROSASCO (2015). « Discovering discrete subword units with binarized autoencoders and hidden-markov-model encoders ». In : *Proc. INTERSPEECH*. IEEE, p. 3174–3178.

BALJEKAR, Pallavi et al. (2015). « Using articulatory features and inferred phonological segments in zero resource speech processing ». In : *Proc. INTERSPEECH*. IEEE, p. 3194–3198.

BARTELS, Chris et al. (2016). « Toward human-assisted lexical unit discovery without text resources ». In : *Proc. SLT*. IEEE, p. 64–70.

BASTIEN, Frédéric et al. (2012). *Theano : new features and speed improvements*. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

BELKIN, Mikhail et Partha NIYOGI (2003). « Laplacian eigenmaps for dimensionality reduction and data representation ». In : *Neural computation* 15.6, p. 1373–1396.

BESACIER, Laurent et al. (2014). « Automatic speech recognition for under-resourced languages : A survey ». In : *Speech Communication* 56, p. 85–100.

BHATI, Saurabhchand, Shekhar NAYAK et K Sri Rama MURTY (2017). « Unsupervised Speech Signal to Symbol Transformation for Zero Resource Speech Applications ». In : IEEE, p. 2133–2137.

BILMES, Jeff A et al. (1998). « A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models ». In : *International Computer Science Institute* 4.510, p. 126.

BLEI, David M, Michael I JORDAN et al. (2006). « Variational inference for Dirichlet process mixtures ». In : *Bayesian analysis* 1.1, p. 121–143.

BOUSQUET, P.-M., D. MATROUF, J.-F. BONASTRE et al. (2011). « Intersession Compensation and Scoring Methods in the i-vectors Space for Speaker Recognition ». In : *Proc. INTERSPEECH*. IEEE, p. 485–488.

BURGET, Lukáš et al. (2010). « Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models ». In : *Proc. ICASSP*. IEEE, p. 4334–4337.

CAI, Meng et al. (2015). « High-performance Swahili keyword search with very limited language pack : The THUEE system for the OpenKWS15 evaluation ». In : *Proc. ASRU*. IEEE, p. 215–222.

CHANG, Jason et John W FISHER III (2013). « Parallel sampling of DP mixture models using sub-cluster splits ». In : *Proc. NIPS*, p. 620–628.

CHEN, Guoguo et al. (2013). « Using proxies for OOV keywords in the keyword search task ». In : *Proc. ASRU*. IEEE, p. 416–421.

CHEN, Hongjie et al. (2015). « Parallel inference of Dirichlet process Gaussian mixture models for unsupervised acoustic modeling : A feasibility study ». In : *Proc. INTERSPEECH*. IEEE, p. 3189–3193.

— (2017). « Multilingual bottle-neck feature learning from untranscribed speech ». In : *Proc. ASRU*. IEEE, p. 727–733.

CHEN, Wen-Yen et al. (2011). « Parallel spectral clustering in distributed systems ». In : *transactions on pattern analysis and machine intelligence* 33.3, p. 568–586.

CHITROUB, Salim (2004). « Analyse en composantes principales d images optiques de télédétection ». In : *Approche neuronale, CARI*, p. 51–58.

CHUNG, Yu-An et al. (2016). « Audio word2vec : Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder ». In : *Proc. INTERSPEECH*. IEEE, p. 765–769.

CHUNG, Cheng-Tao, Chun-an CHAN et Lin-shan LEE (2013). « Unsupervised discovery of linguistic structure including two-level acoustic patterns using three cascaded stages of iterative optimization ». In : *Proc. ICASSP*. IEEE, p. 8081–8085.

CHUNG, Cheng-Tao, Cheng-Yu TSAI, Chia-Hsiang LIU et al. (2017). « Unsupervised Iterative Deep Learning of Speech Features and Acoustic Tokens with Applications to Spoken Term Detection ». In : *ASLP* 25.10, p. 1914–1928.

CHUNG, Cheng-Tao, Cheng-Yu TSAI, Hsiang-Hung LU et al. (2015). « An iterative deep learning framework for unsupervised discovery of speech features and linguistic units with applications on spoken term detection ». In : *Proc. ASRU*. IEEE, p. 245–251.

CHURCH, Kenneth Ward et Jonathan Isaac HELFMAN (1993). « Dotplot : A program for exploring self-similarity in millions of lines of text and code ». In : *Journal of Computational and Graphical Statistics* 2.2, p. 153–174.

COATES, Adam et Andrew Y NG (2012). « Learning feature representations with k-means ». In : *Neural networks : Tricks of the trade*. Springer, p. 561–580.

CUI, Jia et al. (2015). « Multilingual representations for low resource speech recognition and keyword search ». In : *Proc. ASRU*. IEEE, p. 259–266.

CYBENKO, George (1989). « Approximation by superpositions of a sigmoidal function ». In : *Mathematics of control, signals and systems* 2.4, p. 303–314.

DAHL, George E, Tara N SAINATH et Geoffrey E HINTON (2013). « Improving deep neural networks for LVCSR using rectified linear units and dropout ». In : *Proc. ICASSP*. IEEE, p. 8609–8613.

DAVEL, Marelie et al. (2015). « Exploring minimal pronunciation modeling for low resource languages ». In : *Proc. INTERSPEECH*. IEEE, p. 538–542.

DE MULDER, Wim, Steven BETHARD et Marie-Francine MOENS (2015). « A survey on the application of recurrent neural networks to statistical language modeling ». In : *Computer Speech & Language* 30.1, p. 61–98.

DHILLON, Inderjit S, Yuqiang GUAN et Brian KULIS (2004). *A unified view of kernel k-means, spectral clustering and graph cuts*. Citeseer.

DIELEMAN, Sander et al. (2015). *Lasagne : First release*. URL : https://zenodo.org/record/2787.

DING, Bin, Huimin QIAN et Jun ZHOU (2018). « Activation functions and their characteristics in deep neural networks ». In : *Proc. CCDC*. IEEE, p. 1836–1841.

DUCHI, John, Elad HAZAN et Yoram SINGER (2011). « Adaptive subgradient methods for online learning and stochastic optimization ». In : *Journal of Machine Learning Research* 12.Jul, p. 2121–2159.

DUNBAR, Ewan et al. (2017). « The zero resource speech challenge 2017 ». In : IEEE, p. 323–330.

DUTTA, Indranil et Ayushi PANDEY (2015). « Acoustics of articulatory constraints : Vowel classification and nasalization ». In : *Proc. INTERSPEECH*. IEEE, p. 1700–1704.

ERICKSON, Bradley J et al. (2017). « Toolkits and libraries for deep learning ». In : *Journal of digital imaging* 30.4, p. 400–405.

ESCOFIER, Brigitte et Jérôme PAGÈS (2008). *Analyses factorielles simples et multiples : objectifs, méthodes et interprétation*. Dunod, p. 7–30.

FENG, Siyuan et Tan LEE (2018). « Exploiting Speaker and Phonetic Diversity of Mismatched Language Resources for Unsupervised Subword Modeling ». In : IEEE, p. 2673–2677.

FERRAS, Marc et al. (2007). « Constrained MLLR for speaker recognition ». In : *Proc. ICASSP*. T. 4. IEEE, p. IV-53–IV-56.

FREITAG, Michael et al. (2017). « audeep : Unsupervised learning of representations from audio with deep recurrent neural networks ». In : *The Journal of Machine Learning Research* 18.1, p. 6340–6344.

GANAPATHY, Sriram et al. (2015). « Investigating factor analysis features for deep neural networks in noisy speech recognition ». In : *Proc. INTERSPEECH*. IEEE, p. 1898–1902.

GARDNER, Matt W et SR DORLING (1998). « Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences ». In : *Atmospheric environment* 32.14-15, p. 2627–2636.

GAROFOLO, John S (1993). « TIMIT acoustic phonetic continuous speech corpus ». In : *Linguistic Data Consortium*.

GELLY, Gregory et Jean-Luc GAUVAIN (2018). « Optimization of RNN-Based Speech Activity Detection ». In : *ASLP* 26.3, p. 646–656.

GERVEN, Stefaan Van et Fei XIE (1997). « A comparative study of speech detection methods ». In : *Proc. EUROSPEECH*, p. 1095–1098.

GOLDWATER, Sharon, Thomas L GRIFFITHS et Mark JOHNSON (2009). « A Bayesian framework for word segmentation : Exploring the effects of context ». In : *Cognition* 112.1, p. 21–54.

GOLIK, Pavel et al. (2015). « Multilingual features based keyword search for very low-resource languages ». In : *Proc. INTERSPEECH*. IEEE, p. 1260–1264.

GRAVES, Alex, Abdel-rahman MOHAMED et Geoffrey HINTON (2013). « Speech recognition with deep recurrent neural networks ». In : *Proc. ICASSP*. IEEE, p. 6645–6649.

GUPTA, Shruti, Karthik P RAMESH et Erik P BLASCH (2008). « Mutual information metric evaluation for pet/mri image fusion ». In : *Proc. NAECON*. IEEE, p. 305–311.

GWON, Youngjune Lee et al. (2017). « Language recognition via sparse coding ». In : *Proc. INTERSPEECH*. IEEE, p. 2920–2924.

HAEB-UMBACH, Reinhold (1999). « Investigations on inter-speaker variability in the feature space ». In : *Proc. ICASSP*. T. 1. IEEE, p. 397–400.

HALL, Mark A et Lloyd A SMITH (1999). « Feature selection for machine learning : comparing a correlation-based filter approach to the wrapper ». In : *Proc. FLAIRS*. T. 1999. IEEE, p. 235–239.

HALLÉ, Pierre (2004). « Acquisition du langage : spécialisation des enfants dans leur langue maternelle ». In : *Proc. MIDL*, p. 29–30.

HAMPSHIRE, John B et Alex H WAIBEL (1990). « The meta-pi network : Connectionist rapid adaptation for high-performance multi-speaker phoneme recognition ». In : *Proc. ICASSP*. IEEE, p. 165–168.

HANNUN, Awni Y. et al. (2014). « Deep Speech : Scaling up end-to-end speech recognition ». In : *CoRR*.

HAVARD, William (2017). « Découverte non supervisée de lexique à partir d'un corpus multimodal pour la documentation des langues en danger ». Thèse de doct. Sciences de l'Homme et Société.

HECK, Michael, Sakriani SAKTI et Satoshi NAKAMURA (2016a). « Iterative training of a DPGMM-HMM acoustic unit recognizer in a zero resource scenario ». In : *Proc. SLT*. IEEE, p. 57–63.

— (2016b). « Supervised Learning of Acoustic Models in a Zero Resource Setting to Improve DPGMM Clustering ». In : *Proc. INTERSPEECH*. IEEE, p. 1310–1314.

— (2017). « Feature optimized dpgmm clustering for unsupervised subword modeling : A contribution to zerospeech 2017 ». In : *Proc. ASRU*. IEEE, p. 740–746.

— (2018a). « Dirichlet Process Mixture of Mixtures Model for Unsupervised Subword Modeling ». In : *ASLP* 26.11, p. 2027–2042.

— (2018b). « Learning Supervised Feature Transformations on Zero Resources for Improved Acoustic Unit Discovery ». In : *IEICE Transactions on Information and Systems* 101.1, p. 205–214.

HEERDEN, Charl van, Marelie H DAVEL et Etienne BARNARD (2013). « The semi-automated creation of stratified speech corpora ». In : *Proc. PRASA*, p. 115–119.

HELMKE, Hartmut et al. (2015). « Assistant-based speech recognition for ATM applications ». In : *Proc. ATM*.

HINTON, Geoffrey E et Ruslan R SALAKHUTDINOV (2006). « Reducing the dimensionality of data with neural networks ». In : *science* 313.5786, p. 504–507.

HINTON, Geoffrey et al. (2012). « Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups ». In : *Signal processing magazine* 29.6, p. 82–97.

HOANG, Dac-Thang, Tat-Thang VU et Tung-Lam PHI (2016). « Blind method for phone segmentation using Gaussian function locally ». In : *Proc. RIVF*. IEEE, p. 137–141.

HOY, Matthew B (2018). « Alexa, Siri, Cortana, and More : An Introduction to Voice Assistants ». In : *Medical reference services quarterly* 37.1, p. 81–88.

HU, Wenping et al. (2015). « Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers ». In : *Speech Communication* 67, p. 154–166.

HUANG, Gao et al. (2017). « Densely connected convolutional networks ». In : *Proc. CVPR*. T. 1 :2. IEEE, p. 4700–4708.

HUANG, Xuedong, James BAKER et Raj REDDY (2014). « A historical perspective of speech recognition ». In : *Communications of the ACM* 57.1, p. 94–103.

IOFFE, Sergey et Christian SZEGEDY (2015). « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift ». In : *Proc. ICML*. ACM. PMLR, p. 448–456.

JAITLY, Navdeep et Geoffrey E HINTON (2013). « Vocal tract length perturbation (VTLP) improves speech recognition ». In : *Proc. ICML, Workshop on Deep Learning for Audio, Speech and Language*. ACM.

JANSEN, Aren, Kenneth CHURCH et Hynek HERMANSKY (2010). « Towards spoken term discovery at scale with zero resources ». In : *Proc. INTERSPEECH*. IEEE, p. 1676–1679.

JANSEN, Aren, Emmanuel DUPOUX et al. (2013). « A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and models of early language acquisition ». In : *Proc. INTERSPEECH*. IEEE, p. 8111–8115.

JANSEN, Aren, Manoj PLAKAL et al. (2017a). « Towards learning semantic audio representations from unlabeled data ». In : *signal* 2.3, p. 7–11.

— (2017b). « Unsupervised learning of semantic audio representations ». In : *Proc. ICASSP*. IEEE, p. 126–130.

JANSEN, Aren et Benjamin VAN DURME (2011). « Efficient spoken term discovery using randomized algorithms ». In : *Proc. ASRU*. IEEE, p. 401–406.

JEMAI, Olfa et al. (2015). « A speech recognition system based on hybrid wavelet network including a fuzzy decision support system ». In : *Proc. ICMV 2014*. T. 9445. International Society for Optics et Photonics. SPIE, p. 9445.03.

JONES, Douglas et al. (2009). « Speech Processing : Theory of LPC Analysis and Synthesis ». In : *Connexions. June* 1.

KAMPER, Herman (2017). « Unsupervised neural and Bayesian models for zero-resource speech processing ».

KAMPER, Herman, Micha ELSNER et al. (2015). « Unsupervised neural network based feature extraction using weak top-down constraints ». In : *Proc. ICASSP*. IEEE, p. 5818–5822.

KAMPER, Herman, Karen LIVESCU et Sharon GOLDWATER (2017). « An embedded segmental k-means model for unsupervised segmentation and clustering of speech ». In : *Proc. ASRU*. IEEE, p. 719–726.

KESSY, Agnan, Alex LEWIN et Korbinian STRIMMER (2018). « Optimal whitening and decorrelation ». In : *The American Statistician*, p. 1–6.

KIESLING, Scott, Laura DILLEY et William D RAYMOND (2006). « The Variation in Conversation (ViC) Project : Creation of the Buckeye Corpus of Conversational Speech ». In : *Language Variation and Change*, p. 55–97.

KINGMA, Diederik P et Jimmy BA (2015). « Adam : A method for stochastic optimization ». In : *Proc. ICLR*.

KINOSHITA, Keisuke et al. (2009). « Suppression of late reverberation effect on speech signal using long-term multiple-step linear prediction ». In : *ASLP* 17.4, p. 534–545.

KRAMER, Mark A (1991). « Nonlinear principal component analysis using autoassociative neural networks ». In : *AIChE journal* 37.2, p. 233–243.

LAMEL, Lori F. et al. (1993). « BREF, a Large Vocabulary Spoken Corpus for French ». In : *Proc. EUROSPEECH*, p. 505–508.

LARTILLOT, Olivier, Petri TOIVIAINEN et Tuomas EEROLA (2008). « A matlab toolbox for music information retrieval ». In : *Data analysis, machine learning and applications*. Springer, p. 261–268.

LAZARIDIS, Alexandros et al. (2016). *Investigating cross-lingual multi-level adaptive networks : The importance of the correlation of source and target languages*. Rapp. tech.

LECUN, Yann, Yoshua BENGIO et al. (1995). « Convolutional networks for images, speech, and time series ». In : *The handbook of brain theory and neural networks* 3361.10, p. 255–258.

LEE, Chia-ying et James GLASS (2012). « A nonparametric Bayesian approach to acoustic model discovery ». In : *Proc. ACL*. Association for Computational Linguistics, p. 40–49.

LEE, Chia-ying, Timothy J O'DONNELL et James GLASS (2015). « Unsupervised lexicon discovery from acoustic input ». In : *Transactions of the Association for Computational Linguistics* 3, p. 389–403.

LEE, Chin-Hui, Frank K SOONG et Bing-Hwang JUANG (1988). « A segment model based approach to speech recognition ». In : *Proc. ICASSP*. IEEE, p. 501–541.

LEE, K-F et H-W HON (1989). « Speaker-independent phone recognition using hidden Markov models ». In : *Transactions on Acoustics, Speech, and Signal Processing* 37.11, p. 1641–1648.

LERATO, Lerato et Thomas NIESLER (2012). « Investigating parameters for unsupervised clustering of speech segments using TIMIT ». In : *Proc. PRASA*. IEEE, p. 83–88.

— (2015). « Clustering acoustic segments using multi-stage agglomerative hierarchical clustering ». In : *PloS one* 10.10, e0141756.

LI, Jinyu et al. (2018). « Advancing Acoustic-to-Word CTC Model ». In : *Proc. ICASSP*. IEEE, p. 5794–5798.

LIAO, Hank (2013). « Speaker adaptation of context dependent deep neural networks ». In : *Proc. ICASSP*. IEEE, p. 7947–7951.

LIU, Weibo et al. (2017). « A survey of deep neural network architectures and their applications ». In : *Neurocomputing* 234, p. 11–26.

LOISELLE, Stéphane (2004). *Exploration de réseaux de neurones à décharges dans un contexte de reconnaissance de parole*. Université du Québec à Chicoutimi.

LOISELLE, Stéphane et al. (2005). « Exploration of rank order coding with spiking neural networks for speech recognition ». In : *Proc. IJCNN*. T. 4. IEEE, p. 2076–2080.

LU, Liang, Arnab GHOSHAL et Steve RENALS (2011). « Regularized subspace Gaussian mixture models for cross-lingual speech recognition ». In : *Proc. ASRU*. IEEE, p. 365–370.

— (2012). « Maximum a posteriori adaptation of subspace Gaussian mixture models for cross-lingual speech recognition ». In : *Proc. ICASSP*. IEEE, p. 4877–4880.

MANDAL, Anupam, KR Prasanna KUMAR et Pabitra MITRA (2014). « Recent developments in spoken term detection : a survey ». In : *International Journal of Speech Technology* 17.2, p. 183–198.

MANENTI, Céline, Thomas PELLEGRINI et Julien PINQUIER (2016a). « CNN-based phone segmentation experiments in a less-represented language ». In : *Proc. INTERSPEECH*. IEEE, p. 3549–3553.

— (2016b). « Influence de la quantité de données sur une tâche de segmentation de phones fondée sur les réseaux de neurones ». In : *Journees d'Etudes sur la Parole (JEP 2016)*, p. 392–400.

— (2017). « Unsupervised Speech Unit Discovery Using K-means and Neural Networks ». In : *International Conference on Statistical Language and Speech Processing*. Springer, p. 169–180.

— (p.d.). « Unsupervised Representation Learning of Pseudo-phones using Clustering and CNNs ». in revision in Journal : Computer Speech Language.

MANJUNATH, KE, K Sreenivasa RAO et Dinesh Babu JAYAGOPI (2017). « Development of multilingual phone recognition system for Indian languages ». In : *Proc. SPICES*. IEEE, p. 1–6.

MARTIN, Alvin et al. (1997). *The DET curve in assessment of detection task performance*. Rapp. tech. DTIC Document.

MIAO, Yajie (2014). « Kaldi+ PDNN : building DNN-based ASR systems with Kaldi and PDNN ». In : *CoRR*.

MIAO, Yajie, Florian METZE et Alex WAIBEL (2013). « Subspace mixture model for low-resource speech recognition in cross-lingual settings ». In : *Proc. ICASSP*. IEEE, p. 7339–7343.

MICHAUD, Alexis et al. (2016). « Contribuer au progrès solidaire des recherches et de la documentation : la Collection Pangloss et la Collection AuCo ». In : *Proc. JEP*. T. 1, p. 155–163.

MILLER, Kenneth D et David JC MACKAY (1994). « The role of constraints in Hebbian learning ». In : *Neural Computation* 6.1, p. 100–126.

MITRA, Vikramjit, Dimitra VERGYRI et Horacio FRANCO (2016). « Unsupervised Learning of Acoustic Units Using Autoencoders and Kohonen Nets ». In : *Proc. INTERSPEECH*. IEEE, p. 1300–1304.

MIYAZAWA, Kouki et al. (2011). « The Multi Timescale Phoneme Acquisition Model of the Self-Organizing Based on the Dynamic Features ». In : *Proc. INTERSPEECH*. IEEE, p. 749–752.

MOTLICEK, Petr, Fabio VALENTE et Philip N GARNER (2010). « English spoken term detection in multilingual recordings ». In : *Proc. INTERSPEECH*. IEEE, p. 206–209.

MÜLLER, Markus, Sebastian STÜKER et Alex WAIBEL (2018). « Neural Language Codes for Multilingual Acoustic Models ». In : *Proc. INTERSPEECH*. IEEE, p. 2419–2423.

MUMTAZ, Benazir et al. (2014). « Multitier annotation of Urdu speech corpus ». In : *Proc. CLT*, p. 47–54.

MURTAGH, Fionn et Pierre LEGENDRE (2014). « Ward's hierarchical agglomerative clustering method : which algorithms implement Ward's criterion ? » In : *Journal of classification* 31.3, p. 274–295.

MUTHUSAMY, Yeshwant K, Ronald A COLE et Beatrice T OSHIKA (1992). « The OGI multi-language telephone speech corpus ». In : *Proc. ICSLP*. IEEE, p. 105–113.

NIST (2013). *The OpenKWS14 Evaluation Plan*. URL : https://www.nist.gov/sites/default/files/documents/itl/iad/mig/KWS14-evalplan-v11.pdf.

NIST Open Keyword Search Evaluation (OpenKWS) (2013). URL : https://www.nist.gov/itl/iad/mig/open-keyword-search-evaluation.

ONDEL, Lucas, Lukáš BURGET et Jan ČERNOCK (2016). « Variational inference for acoustic unit discovery ». In : *Procedia Computer Science* 81, p. 80–86.

ONDEL, Lucas, Lukaš BURGET et al. (2017). « Bayesian phonotactic language model for acoustic unit discovery ». In : *Proc. ICASSP*. IEEE, p. 5750–5754.

OOSTDIJK, NHJ et al. (2002). « Experiences from the spoken Dutch corpus project ». In : *Proc. LREC*, p. 340–347.

PALAZ, Dimitri, Ronan COLLOBERT et Mathew Magimai DOSS (2013). « Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks ». In : *Proc. INTERSPEECH*. IEEE, p. 1766–1770.

PALAZ, Dimitri, Ronan COLLOBERT et al. (2015). « Analysis of cnn-based speech recognition system using raw speech as input ». In : *Proc. INTERSPEECH*. IEEE, p. 11–15.

PARK, Alex S et James R GLASS (2008). « Unsupervised pattern discovery in speech ». In : *ASLP* 16.1, p. 186–197.

PARTHASARATHI, Sree Hari Krishnan et al. (2015). « fMLLR based feature-space speaker adaptation of DNN acoustic models ». In : *Proc. INTERSPEECH*. IEEE, p. 3630–3634.

PEDREGOSA, F. et al. (2011). « Scikit-learn : Machine Learning in Python ». In : *Journal of Machine Learning Research* 12, p. 2825–2830.

PELLEGRINI, Thomas (2008). « Transcription automatique de langues peu dotées ». Thèse de doct. Université Paris Sud-Paris XI.

PELLEGRINI, Thomas, Céline MANENTI et Julien PINQUIER (2017). *The IRIT-UPS system@ ZeroSpeech 2017 Track1 : unsupervised subword modeling*. Rapp. tech. UPS, IRIT.

PITT, Mark A et al. (2005). « The Buckeye corpus of conversational speech : labeling conventions and a test of transcriber reliability ». In : *Speech Communication* 45.1, p. 89–95.

PITT, M.A. et al. (2007). *Buckeye Corpus of Conversational Speech (2nd release)*. URL : www.buckeyecorpus.osu.edu.

POVEY, Daniel, Lukáš BURGET et al. (2011). « The subspace Gaussian mixture model—A structured model for speech recognition ». In : *Computer Speech & Language* 25.2, p. 404–439.

POVEY, Daniel, Lukśš BURGET et al. (2010). « Subspace Gaussian mixture models for speech recognition ». In : *Proc. ICASSP*. IEEE, p. 4330–4333.

POVEY, Daniel, Arnab GHOSHAL et al. (2011). « The Kaldi speech recognition toolkit ». In : *Proc. ASRU*. IEEE, p. 786–789.

QIAN, Ning (1999). « On the momentum term in gradient descent learning algorithms ». In : *Neural networks* 12.1, p. 145–151.

QIAO, Yu, Naoya SHIMOMURA et Nobuaki MINEMATSU (2008). « Unsupervised optimal phoneme segmentation : Objectives, algorithm and comparisons ». In : *Proc. ICASSP*. IEEE, p. 3989–3992.

RABINER, Lawrence R (1989). « A tutorial on hidden Markov models and selected applications in speech recognition ». In : *Proceedings of the IEEE* 77.2, p. 257–286.

RAYMOND, William D et al. (2002). « An analysis of transcription consistency in spontaneous speech from the Buckeye corpus ». In : *Proc. ICSLP*. IEEE, p. 1125–1128.

RECASENS, Daniel, Maria Dolors PALLARÈS et Jordi FONTDEVILA (1997). « A model of lingual coarticulation based on articulatory constraints ». In : *The Journal of the Acoustical Society of America* 102.1, p. 544–561.

RENSHAW, Daniel et al. (2015). « A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge ». In : *Proc. INTERSPEECH*. IEEE, p. 3199–3203.

RIAD, Rachid et al. (2018). « Sampling strategies in Siamese Networks for unsupervised speech representation learning ». In : *Proc. INTERSPEECH*. IEEE, p. 2658–2662.

RODRIGUEZ, Alex et Alessandro LAIO (2014). « Clustering by fast search and find of density peaks ». In : *Science* 344.6191, p. 1492–1496.

ROSENBLATT, Frank (1958). « The perceptron : a probabilistic model for information storage and organization in the brain ». In : *Psychological review* 65.6, p. 386–408.

ROUVIER, Mickael et al. (2013). « An open-source state-of-the-art toolbox for broadcast news diarization ». In : *Proc. INTERSPEECH*. IEEE, p. 1477–1481.

RUDER, Sebastian (2016). « An overview of gradient descent optimization algorithms ». In : *arXiv preprint arXiv :1609.04747*.

SAK, Haşim et al. (2015). « Fast and accurate recurrent neural network acoustic models for speech recognition ». In : *Proc. INTERSPEECH*. IEEE, p. 1468–1472.

SAKOE, Hiroaki et Seibi CHIBA (1978). « Dynamic programming algorithm optimization for spoken word recognition ». In : *transactions on acoustics, speech, and signal processing* 26.1, p. 43–49.

SAMSON, Sarah et al. (2015). « Using resources from a closely-related language to develop ASR for a very under-resourced language : A case study for Iban ». In : *Proc. INTERSPEECH*. IEEE, p. 1270–1274.

SAON, George et Jen-Tzung CHIEN (2012). « Large-vocabulary continuous speech recognition systems : A look at some recent advances ». In : *Signal Processing Magazine* 29.6, p. 18–33.

SAON, George, Geoffrey ZWEIG et Mukund PADMANABHAN (2001). « Linear feature space projections for speaker adaptation ». In : *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. T. 1. IEEE, p. 325–328.

SCHARENBORG, Odette, Francesco CIANNELLA et al. (2017). « Building an asr system for a low-resource language through the adaptation of a high-resource language asr system : Preliminary results ». In : *Proc. ICNLSSP*.

SCHARENBORG, Odette, Vincent WAN et Mirjam ERNESTUS (2010). « Unsupervised speech segmentation : An analysis of the hypothesized phone boundaries ». In : *The Journal of the Acoustical Society of America* 127.2, p. 1084–1095.

SCHATZ, Thomas, Vijayaditya PEDDINTI, Francis BACH et al. (2013). « Evaluating speech features with the minimal-pair ABX task : Analysis of the classical MFC/PLP pipeline ». In : *Proc. INTERSPEECH*. IEEE, p. 1–5.

SCHATZ, Thomas, Vijayaditya PEDDINTI, Xuan-Nga CAO et al. (2014). « Evaluating speech features with the Minimal-Pair ABX task (II) : Resistance to noise ». In : *Proc. INTERSPEECH*. IEEE, p. 915–919.

SCHAUL, Tom et al. (2010). « PyBrain ». In : *Journal of Machine Learning Research* 11.Feb, p. 743–746.

SCHMIDHUBER, Jürgen (2015). « Deep learning in neural networks : An overview ». In : *Neural networks* 61, p. 85–117.

SCHULTZ, Tanja et Tim SCHLIPPE (2014). « GlobalPhone : Pronunciation Dictionaries in 20 Languages ». In : *Proc. LREC*, p. 337–341.

SCHULTZ, Tanja et Alex WAIBEL (2001). « Language-independent and language-adaptive acoustic modeling for speech recognition ». In : *Speech Communication* 35.1-2, p. 31–51.

SCHULTZ, Tanja, Martin WESTPHAL et Alex WAIBEL (1997). « The globalphone project : Multilingual lvcsr with janus-3 ». In : *Multilingual Information Retrieval Dialogs : 2nd SQEL Workshop*, p. 20–27.

SCHUURMAN, Ineke et al. (2004). « Linguistic Annotation of the Spoken Dutch Corpus : If We Had To Do It All Over Again ». In : *Proc. LREC*, p. 57–60.

SHARMA, Pulkit et al. (2018). « Sparse coding based features for speech units classification ». In : *Computer Speech & Language* 47, p. 333–350.

SINISCALCHI, Sabato Marco, Dau-Cheng LYU et al. (2012). « Experiments on cross-language attribute detection and phone recognition with minimal target-specific training data ». In : *ASLP* 20.3, p. 875–887.

SINISCALCHI, Sabato Marco, Torbjorn SVENDSEN et Chin-Hui LEE (2008). « Toward a detector-based universal phone recognizer ». In : *Proc. ICASSP*. IEEE, p. 4261–4264.

SIU, Man-hung et al. (2014). « Unsupervised training of an HMM-based self-organizing unit recognizer with applications to topic classification and keyword discovery ». In : *Computer Speech & Language* 28.1, p. 210–223.

SOLTAU, Hagen, Hank LIAO et Hasim SAK (2016). « Neural speech recognizer : Acoustic-to-word LSTM model for large vocabulary speech recognition ». In : IEEE, p. 3707–3711.

SRIKANTHAN, Sharanyan, Arvind KUMAR et Rajeev GUPTA (2011). « Implementing the dynamic time warping algorithm in multithreaded environments for real time and unsupervised pattern discovery ». In : *Proc. ICCCT*. IEEE, p. 394–398.

SRIRANJANI, R, S UMESH et al. (2015). « Investigation of different acoustic modeling techniques for low resource indian language data ». In : *Proc. NCC*. IEEE, p. 1–5.

STADTSCHNITZER, Michael et Christoph SCHMIDT (2015). « Implementation of a Live Dialectal Media Subtitling System ». In : *Proc. INTERSPEECH*. IEEE, p. 728–729.

STAHLBERG, Felix et al. (2014). « Towards automatic speech recognition without pronunciation dictionary, transcribed speech and text resources in the target language using cross-lingual word-to-phoneme alignment ». In : *Proc. SLTU*. IEEE, p. 73–80.

STOLCKE, Andreas et al. (2006). « Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons ». In : *Proc. ICASSP*. T. 1. IEEE, p. I-321–I-324.

SUTSKEVER, Ilya, Oriol VINYALS et Quoc V LE (2014). « Sequence to sequence learning with neural networks ». In : *Proc. NIPS*, p. 3104–3112.

SWIETOJANSKI, Pawel et Steve RENALS (2014). « Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models ». In : *Proc. SLST*. IEEE, p. 171–176.

SYNNAEVE, Gabriel et Emmanuel DUPOUX (2015). « Weakly supervised multi-embeddings learning of acoustic models ». In : *Proc. ICLR*, p. 95–100.

— (2016). « A temporal coherence loss function for learning unsupervised acoustic embeddings ». In : *Procedia Computer Science* 81, p. 95–100.

SYNNAEVE, Gabriel, Thomas SCHATZ et Emmanuel DUPOUX (2014). « Phonetics embedding learning with side information ». In : *Proc. SLT*. IEEE, p. 106–111.

SZOKE, Igor, Lukás BURGET et al. (2008). « Sub-word modeling of out of vocabulary words in spoken term detection ». In : *Proc. SLT*. IEEE, p. 273–276.

SZOKE, Igor, Petr SCHWARZ et al. (2005). « Comparison of keyword spotting approaches for informal continuous speech ». In : *Proc. EUROSPEECH*. IEEE, p. 633–636.

TANIGUCHI, Tadahiro et al. (2016). « Double articulation analyzer with deep sparse autoencoder for unsupervised word discovery from speech signals ». In : *Advanced Robotics* 30.11-12, p. 770–783.

TAVANAEI, Amirhossein et Anthony S MAIDA (2017). « A spiking network that learns to extract spike signatures from speech signals ». In : *Neurocomputing* 240, p. 191–199.

TEAM, Theano Development (2016). « Theano : A Python framework for fast computation of mathematical expressions ». In : *arXiv e-prints* abs/1605.02688. URL : http://arxiv.org/abs/1605.02688.

The IARPA ASPIRE challenge (2015). URL : http://www.iarpa.gov/index.php/working-with-iarpa/prize-challenges/306-automatic-speech-in-reverberant-environments-aspire-challenge/.

THEIS, Lucas et al. (2017). « Lossy image compression with compressive autoencoders ». In : *ICLR*, p. 2027–2042.

THIOLLIERE, Roland et al. (2015). « A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling ». In : *Proc. INTERSPEECH*. IEEE, p. 3179–3183.

TIAN, Fei et al. (2014). « Learning deep representations for graph clustering ». In : *AAAI*, p. 1293–1299.

TIELEMAN, Tijmen et Geoffrey HINTON (2012). « rmsprop : Divide the gradient by a running average of its recent magnitude ».

TORFI, Amirsina (2017). *SpeechPy : Speech recognition and feature extraction*. URL : https://zenodo.org/record/840395.

— (2018). « Speechpy-a library for speech processing and recognition ». In : *arXiv preprint arXiv :1803.01094*.

TORRE FRADE, Fernando De la (2008). *A Least-Squares Unified View of PCA, LDA, CCA and Spectral Graph Methods*. Rapp. tech. Pittsburgh, PA : Carnegie Mellon University.

TRAVADI, Ruchir et Shrikanth S NARAYANAN (2015). « Ensemble of Gaussian Mixture Localized Neural Networks with Application to Phone Recognition ». In : *Proc. INTERSPEECH*. IEEE, p. 1903–1907.

TRMAL, Jan et al. (2017). « The kaldi openkws system : Improving low resource keyword search ». In : ISCA, p. 3597–3601.

UMEDA, Noriko (1975). « Vowel duration in american english ». In : *The Journal of the Acoustical Society of America* 58.2, p. 434–445.

— (1977). « Consonant duration in American English ». In : *The Journal of the Acoustical Society of America* 61.3, p. 846–858.

VASILESCU, Ioana, Camille DUTREY et Lori LAMEL (2015). « Large scale data based linguistic investigations using speech technology tools : The case of Romanian ». In : *Proc. SpeD*. IEEE, p. 1–6.

VERGIN, Rivarol, Douglas O'SHAUGHNESSY et Azarshid FARHAT (1999). « Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition ». In : *Transactions on speech and audio processing* 7.5, p. 525–532.

VERSTEEGH, Maarten, Xavier ANGUERA et al. (2016). « The zero resource speech challenge 2015 : Proposed approaches and results ». In : *Procedia Computer Science* 81, p. 67–72.

VERSTEEGH, Maarten, Roland THIOLLIERE et al. (2015). « The zero resource speech challenge 2015 ». In : *Proc. INTERSPEECH*. IEEE.

VESELY, Karel, Mirko HANNEMANN et Lukas BURGET (2013). « Semi-supervised training of deep neural networks ». In : *Proc. ASRU*. IEEE, p. 267–272.

VINCENT, Pascal et al. (2008). « Extracting and composing robust features with denoising autoencoders ». In : *Proc. ICML*. ACM, p. 1096–1103.

WANG, Haipeng, Tan LEE et Cheung-Chi LEUNG (2011). « Unsupervised spoken term detection with acoustic segment model ». In : *Proc. Oriental COCOSDA*. IEEE, p. 106–111.

WANG, Haipeng, Tan LEE, Cheung-Chi LEUNG, Bin MA et al. (2013). « Unsupervised mining of acoustic subword units with segment-level Gaussian posteriorgrams ». In : *Proc. INTERSPEECH*. IEEE, p. 2297–2301.

— (2014). « A graph-based Gaussian component clustering approach to unsupervised acoustic modeling ». In : *Proc. INTERSPEECH*. IEEE, p. 875–879.

— (2015). « Acoustic segment modeling with spectral clustering methods ». In : *Proc. TASLP* 23.2, p. 264–277.

WANG, Haipeng, Cheung-Chi LEUNG et al. (2012). « An acoustic segment modeling approach to query-by-example spoken term detection ». In : *Proc. ICASSP*. IEEE, p. 5157–5160.

WEGMANN, Steven et al. (1996). « Speaker normalization on conversational telephone speech ». In : *Proc. ICASSP*. IEEE, p. 339–341.

Welcome to Spectral Python (SPy) (p.d.). URL : http://www.spectralpython.net/.

XU, Haihua, Tze Yuang CHONG et al. (2015). « On the study of very low-resource language keyword search ». In : *Proc. APSIPA*. IEEE, p. 358–364.

XU, Haihua, Hang SU et al. (2016). « Semi-Supervised and Cross-Lingual Knowledge Transfer Learnings for DNN Hybrid Acoustic Models Under Low-Resource Conditions ». In : *Proc. INTERSPEECH*. IEEE, p. 1315–1319.

XU, Hainan et al. (2015). « Modeling phonetic context with non-random forests for speech recognition ». In : *Proc. INTERSPEECH*. IEEE, p. 2117–2121.

YOSHIOKA, Takuya, Anton RAGNI et Mark JF GALES (2014). « Investigation of unsupervised adaptation of DNN acoustic models with filter bank input ». In : *Proc. ICASSP*. IEEE, p. 6344–6348.

YU, Jia et al. (2015). « A density peak clustering approach to unsupervised acoustic subword units discovery ». In : *Proc. APSIPA*. IEEE, p. 178–183.

YUAN, Yougen et al. (2017). « Extracting bottleneck features and word-like pairs from untranscribed speech for feature representation ». In : *Proc. ASRU*. IEEE, p. 734–739.

ZEGHIDOUR, Neil et al. (2016). « A deep scattering spectrum—deep siamese network pipeline for unsupervised acoustic modeling ». In : *Proc. ICASSP*. IEEE, p. 4965–4969.

ZEILER, Matthew D. (2012). « ADADELTA : An Adaptive Learning Rate Method ». In : *CoRR*.

ZHANG, Qinghua et Albert BENVENISTE (1992). « Wavelet networks ». In : *transactions on Neural Networks* 3.6, p. 889–898.

ZHENG, Xin et al. (2014). « Contrastive auto-encoder for phoneme recognition ». In : *Proc. ICASSP*. IEEE, p. 2529–2533.