Big deep voice : indexation de données massives de parole grâce à des réseaux de neurones profonds

Antoine Perquin

Stage de Master Recherche

Rapport de stage

Big deep voice : indexation de données massives de parole grâce à des réseaux de neurones profonds

Domaines : Calcul et langage, Informatique neuronale et évolutionnaire

Auteur :
Antoine Perquin

Superviseurs :
Gwénolé Lecorvé
Damien Lolive
Expression - IRISA

ENSSAT
LANNION

Abstract: Les systèmes de synthèse de parole sont des outils permettant de générer un signal de parole correspondant à un texte. Les solutions actuelles fournissent un signal de qualité, mais la parole générée est peu expressive, notamment en raison de leur jeu de données limité. Pour résoudre ce problème, il est donc nécessaire d'augmenter la quantité de données pour y intégrer de la variabilité. Cela soulève néanmoins plusieurs problématiques : « comment décrire et comparer les données ? » et « comment rechercher des données lorsque leurs descripteurs sont de grande taille et que le nombre de données est conséquent ? ». Le but de ce stage est de mettre en place une mesure de similarité entre phonèmes, puis éventuellement un algorithme de recherche de plus proches voisins, qui pourront être intégrés au système de synthèse de parole de l'équipe Expression. Pour répondre à la problématique de description et comparaison des données, nous utiliserons la propriété de plongement des réseaux de neurones. Pour répondre à la problématique de recherche de données, nous proposons d'utiliser une méthode de recherche de plus proches voisins en grande dimension. Le but de ce stage est d'étudier la faisabilité de la solution choisie, notamment en proposant des méthodes d'évaluation de la qualité d'un plongement.

Table des matières

Introduction

La synthèse de parole à partir de texte est un domaine qui possède de nombreuses applications. Actuellement, elle permet par exemple de réaliser des interactions humain-machine, notamment sur des serveurs de service clientèle. Elle permet aussi de prêter assistance aux personnes handicapées en permettant la lecture sur ordinateur pour les personnes malvoyantes ou l'expression orale pour les personnes atteintes de troubles de la parole. La synthèse de parole étant un sujet d'étude depuis le XVIII^{e} siècle, de nombreuses solutions existent déjà et permettent d'obtenir une parole compréhensible. Cependant, l'expressivité des signaux produits n'est toujours pas suffisante pour des applications plus ambitieuses telles que le doublage de films ou la lecture d'*audiobooks*. Ainsi, l'ensemble des recherches actuelles cherchent à améliorer les techniques déjà existantes afin d'obtenir des résultats plus expressifs, c'est-à-dire de la parole capable d'exprimer des émotions telles que la joie ou la colère, mais aussi capable d'imiter un accent ou d'adopter un style d'élocution particulier. Pour cela, de nombreux travaux s'appuient sur les avancées récentes dans le domaine de l'apprentissage automatique.

Le cadre et objectif à long terme de ce stage consiste à adapter une technique dite par sélection d'unités afin de permettre l'utilisation de larges collections de paroles pré-enregistrée. Plus précisément, il s'agit de décrire des segments (on parle aussi d'unités) de parole grâce à la propriété de plongement des réseaux de neurones, puis d'effectuer une recherche de plus proches voisins (éventuellement approximative) afin de sélectionner les unités voulues. Dans ce stage, on s'intéresse particulièrement à l'étude des plongements, des distances associées à ces plongements, ainsi qu'à leur évaluation dans le cadre d'une application de synthèse par sélection d'unités. Dans la suite, pour désigner le phénomène de plongement on utilisera le terme d'*embedding* tant il est admis dans la communauté francophone.

Ce rapport débute par un état de l'art présentant différentes solutions de synthèse de parole avant de s'intéresser aux réseaux de neurones et particulièrement leur propriété d'*embeddings* puis les méthodes de recherches de plus proches voisins, en particulier celles approximatives. On pourra alors présenter la contribution proposée par ce stage : différentes méthodes pour évaluer la qualité d'un *embedding* dans le cas d'une synthèse par sélection d'unités guidée par réseau de neurones. Il s'agit tout d'abord de présenter le jeu de données utilisé ainsi que les attributs extraits pour entraîner les réseaux de neurones. On présente ensuite les différents types de réseaux envisagés, leur méthode d'entraînement et d'évaluation. On pourra enfin présenter la contribution de ce stage : la définition de deux méthodes d'évaluation d'*embedding*, l'une basée sur la définition d'un *embedding* pour un espace métrique, l'autre sur la conservation des plus proches voisins.

1 État de l'art

Cet état de l'art a pour but de présenter les éléments clés de la solution de synthèse mise en place lors de ce stage, une synthèse par sélection d'unités guidée par réseaux de neurones. Cet état de l'art présente donc les différentes solutions de synthèse de parole les plus utilisées, en particulier les méthodes par sélection d'unités. On s'intéresse ensuite aux réseaux de neurones et leur propriété d'*embedding* avant d'examiner différentes méthodes de recherche de plus proches voisins approximatives.

1.1 Synthèse de la parole

De manière générale, un système de synthèse de parole se décompose en deux parties (*cf.* figure 1). La première se charge d'analyser et de traiter le texte en entrée du système, analyse linguistique et prosodique sur le schéma. La deuxième partie réalise concrètement la génération du signal de parole à partir de texte, grâce aux informations issues de l'analyse en amont et au moteur de synthèse. Cette section a pour but de définir ce qu'est un système de synthèse ainsi que de présenter les différentes solutions actuelles en matière de moteur, introduction nécessaire aux sections 1.2 et 1.3, plus centrales vis-à-vis des problématiques étudiées. Nous commencerons par présenter le fonctionnement de l'analyse linguistique avant d'examiner différentes solutions de moteurs.

1.1.1 Analyse linguistique et prosodique

Un moteur de synthèse ne peut pas prendre directement du texte en entrée, il nécessite une liste de phonèmes accompagnés éventuellement de consignes telles que la prosodie afin de réaliser la synthèse de parole. Il est donc nécessaire d'effectuer une étape préalable de conversion du texte vers des phonèmes, c'est le rôle de l'analyse linguistique et prosodique. L'analyse fonctionne en trois étapes ([1], chap. 19) : le traitement du texte, l'analyse phonétique et l'analyse prosodique. Contrairement au moteur de synthèse, l'analyse est dépendante de la langue visée par le système de synthèse.

Lors du traitement du texte, celui-ci est d'abord analysé afin d'en identifier la structure : la ponctuation est considérée et le texte est séparé en phrases ou paragraphes. Il est ensuite normalisé, afin de tenir compte de particularités comme les abréviations et les acronymes. Par exemple, en anglais, *St.* peut être interprété *Street* ou *Saint*. Une étape de balisage du texte peut être effectuée afin de modifier la manière dont une séquence doit être interprétée : une suite de nombre séparés par des points peut être un numéro de téléphone ou une adresse IP. Enfin, la dernière étape du

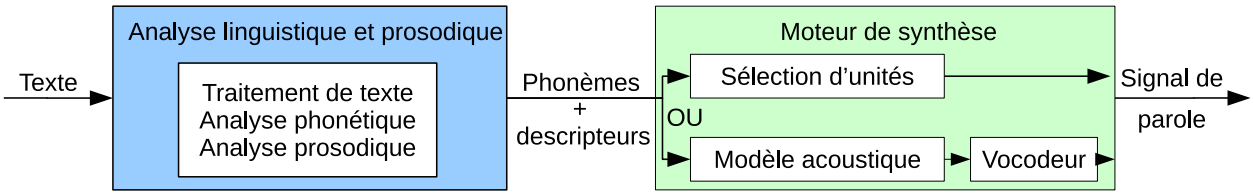

FIGURE 1 – Un système de synthèse permet de convertir un texte en un signal de parole et contient deux blocs fonctionnels : l'analyse linguistique et prosodique et le moteur de synthèse.

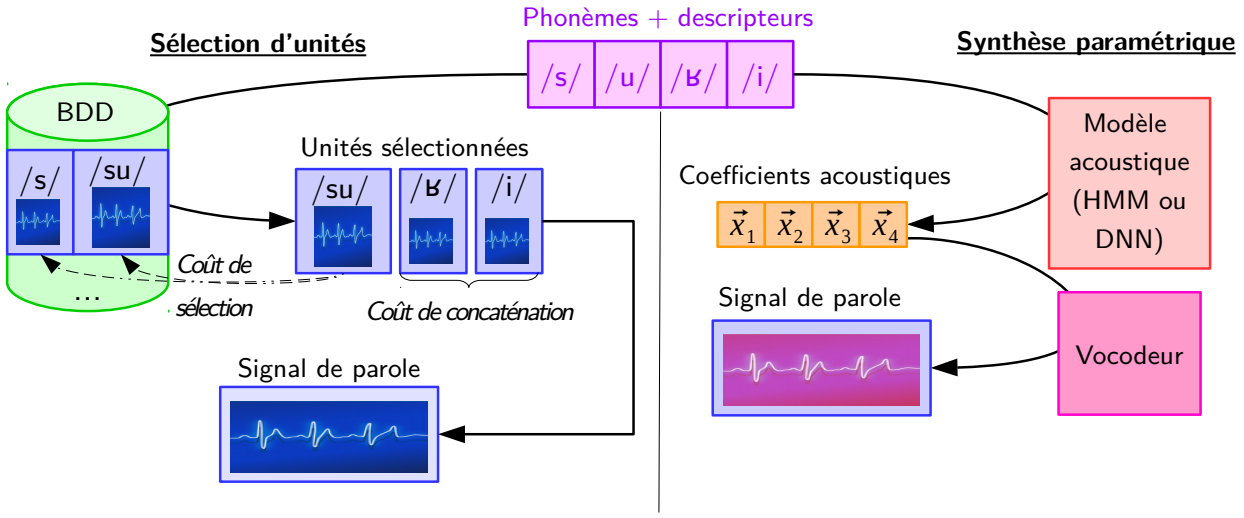

FIGURE 2 – La synthèse par sélection d'unités s'effectue en concaténant des segments de paroles pré-enregistrés dans une base de données tandis que la synthèse paramétrique s'effectue par un vocodeur dont les paramètres sont prédits par un modèle acoustique.

traitement effectue une analyse syntaxique et linguistique qui permet de différencier les homonymes comme « parent » qui peut être lu comme un nom ou un verbe conjugué. Une fois le traitement du texte terminé, chaque mot est identifié. On peut alors procéder à l'étape d'analyse phonétique qui transcrit chaque mot en une séquence de phonèmes. Le plus souvent, cette étape est effectuée en parcourant des dictionnaires de prononciation. Enfin, l'analyse prosodique détermine l'intonation de la parole à générer : intensité, hauteur et durée de chaque phonème. Cette analyse peut par exemple aider à distinguer à l'écoute une question d'une affirmation.

Une fois prédits les phonèmes et la manière dont ils doivent être prononcés, le moteur de synthèse cherche à produire le signal de parole correspondant. Nous allons approfondir ici cette partie en examinant les solutions dites par sélection d'unités et celles fondées sur des modèles statistiques avant d'examiner les solutions hybrides.

1.1.2 Concaténation d'unités

Comme présenté sur la partie gauche de la figure 2, les moteurs de synthèse par concaténation d'unités consistent à mettre bout à bout des morceaux de parole afin de former la phrase souhaitée. Pour cela, on suppose disposer d'une base de données contenant des unités de parole prédécoupées. Ces unités peuvent être de différentes tailles : phonèmes, diphones [1], syllabes, mots ou groupes de mots.

Historiquement, les premiers systèmes par concaténation d'unités étaient ceux par diphones. Pour ces systèmes, les seules unités à disposition dans la base de données sont des diphones enregistrés séparément, parcourant l'ensemble des diphones d'une langue mais chacun en un unique exemplaire. S'en tenir à la concaténation pure de diphones donne cependant des résultats non naturels. On peut alors procéder à des transformations du signal, par exemple via l'algorithme TD-

1. Unité commençant au milieu d'un phonème et s'arrêtant au milieu du phonème suivant.

PSOLA afin de modifier les paramètres prosodiques [2]. Bien que les systèmes de concaténation par diphones présentent l'avantage d'être légers (l'ensemble des unités correspond à seulement quelques minutes d'enregistrement), la parole synthétisée manque toujours de naturel et d'expressivité, ce qui a conduit à la construction de moteurs plus élaborés dits par sélection d'unités.

L'idée de la synthèse par sélection d'unités [3] est de ne pas limiter la base de données à une seule occurrence de chaque diphone. La base contient alors de nombreuses phrases dont le flux de parole a été segmenté en unités de tailles différentes. Par opposition avec les moteurs par concaténation de diphones, il y a plusieurs occurrences de chacun d'entre eux dans la base et chaque occurrence correspond à un contexte de prononciation différent, par exemple avec une prosodie et un placement différent dans la phrase. De plus, la base peut contenir des unités de tailles différentes comme des moitiés de phones ou des mots entiers. Il ne convient plus alors simplement de sélectionner le diphone demandé, mais de choisir l'unité ou l'ensemble d'unités qui donne le meilleur résultat. En effet, la synthèse du mot "chat" peut s'effectuer directement à partir de la sélection du mot lui-même s'il est présent dans la base ou à partir des phonèmes /ʃ/ (**ch**at) et /a/ (pl**a**t) si seules ces unités sont présentes. Cependant, si la base est lacunaire, il faudra parfois se rabattre sur le phonème /ɑ/ (p**â**te) plutôt que /a/ (pl**a**t). La sélection d'unités revient donc à résoudre un problème d'optimisation exprimé comme la recherche de la séquence d'unités minimisant la somme de deux coûts. D'une part, un coût cible permet de choisir les unités les plus proches de celles décrites par l'analyse linguistique. Ce coût est en général exprimé comme la somme pondérée des différences entre les attributs des unités sélectionnées et demandées, la liste et les importances relatives de ces attributs ayant été définie de manière experte. D'autre part, un coût de jointure permet de caractériser si les unités se concatènent bien entre elles, sans donner d'artefacts.

Au final, les systèmes de synthèse fondés sur la sélection d'unités permettent d'obtenir une parole synthétisée de qualité et expressive relativement à leur jeu de données, ce qui explique leur utilisation dans la plupart des solutions commerciales actuelles. Cependant, le coût cible est difficile à exprimer, nécessitant l'aide de linguistes pour prioriser l'importance des descripteurs et les performances de ces systèmes diminuent lorsque qu'aucune unité proche de celle cherchée ne se trouve dans la base. Une solution potentielle à ces problèmes se trouve dans l'utilisation de modèles statistiques paramétriques.

1.1.3 Synthèse statistique paramétrique

Comme présenté sur la partie droite de la figure 2, la synthèse statistique paramétrique consiste à utiliser un modèle acoustique pour prédire, à partir des phonèmes produits par l'analyse linguistique, des coefficients acoustiques qui serviront d'entrée à un vocodeur générant le signal de sortie. Pour apprendre un tel modèle acoustique, il faut disposer d'un large corpus de parole et de sa transcription phonétique. La transformation du corpus sous forme de coefficients acoustiques permet alors l'apprentissage d'un modèle liant phonèmes et ces mêmes coefficients. Ce modèle peut prendre la forme d'un modèle de Markov caché ou d'un réseau de neurone.

Si l'on considère la parole comme une suite d'observations acoustiques correspondant aux phonèmes réalisés, le modèle acoustique utilisé peut être vu comme un modèle de Markov caché (*Hidden Markov Model*, HMM). On parle alors de système HTS (*HMM-based speech synthesis*) [4]. En réalité, ce HMM est construit dynamiquement, à la volée, à chaque utilisation du système de synthèse. Par ailleurs, ce HMM est utilisé de manière non usuelle : on l'utilise pour prédire la séquence d'observations acoustiques correspondant aux phonèmes (et non l'inverse comme d'ordinaire). Pour une phrase à synthétiser, chaque phonème est modélisé par trois états, chacun étant

automatiquement sélectionné à partir d'un ensemble d'états préalablement entraînés. Une fois le modèle construit, les coefficients sont générés trame par trame en cherchant à maximiser la probabilité de la séquence d'états.

Lorsque le modèle acoustique d'un système de synthèse est un réseau de neurone profond (*Deep Neural Network*, DNN), l'entrée du modèle correspond au phonème à synthétiser et à d'autres éventuelles informations générées par l'analyse linguistique. À nouveau, sa sortie correspond aux paramètres attendus par le vocodeur. La durée de chaque phonème peut être calculée en rajoutant une sortie au DNN, ce qui revient à entraîner un DNN multi-tâches dont la tâche principale est de prédire les paramètres acoustiques et la tâche secondaire de prédire la durée du phonème. Les modèles acoustiques fondés sur des DNN s'avèrent souvent plus efficaces que les solutions utilisant des HMM [5]. Certaines extensions de ce principe proposent même de se passer de vocodeur pour prédire directement la forme d'onde du signal [6].

Les systèmes statistiques paramétriques permettent en général de synthétiser une voix relativement expressive, même sur des bases de données plus petites qu'en sélection d'unités. Malheureusement, l'utilisation d'un vocodeur produit parfois un son étouffé. Les systèmes hybrides proposent des éléments de solutions intéressants pour concilier les intérêts de chaque approche.

1.1.4 Systèmes hybrides

Les solutions dites hybrides sont des moteurs de synthèse n'étant ni purement par concaténation ni purement statistiques paramétriques. Par exemple, une méthode par concaténation où une solution de construction des unités manquantes dans la base de données est mise en place [7] est considérée hybride. Comme dans [8], une autre proposition consiste à faire une sélection d'unités guidée par un réseau de neurones. Plus précisément, un DNN est entraîné en tant que modèle acoustique et est utilisé pour définir le coût cible d'un système par sélection d'unités en comparant les attributs prédits par le DNN à ceux des unités présentent dans la base de données. Les résultats des tests d'écoute montrent que cette approche est meilleure qu'une méthode statistique pure. La même expérience effectuée en remplaçant le DNN par un HMM ne permet pas de montrer une amélioration significative de l'approche par DNN par rapport à celle par HMM. On peut cependant penser que ces résultats sont améliorables en utilisant d'autres types de DNN ou en tirant mieux parti de ses propriétés d'*embeddings*, comme nous chercherons à le montrer dans la section suivante.

Ainsi, dans le domaine de la synthèse de parole, deux types de méthodes se distinguent, celles par sélection d'unités et celles statistiques paramétriques. Les solutions par sélection d'unités sont celles qui permettent actuellement d'offrir la meilleure qualité. Cependant, cette méthode offre une expressivité limitée à celle contenue dans son jeu de données. Augmenter l'expressivité d'une solution par concaténation nécessite donc d'augmenter la quantité et la variabilité des données, ce qui pose deux problèmes majeurs. Il faut tout d'abord être capable de correctement décrire et comparer les unités de parole, ce qui est rendu possible par des *embeddings*, présentés dans la section 1.2. De plus, il faut dans le cadre d'une application interactive telle que la synthèse de parole, être algorithmiquement efficace : des algorithmes de recherche de plus proches voisins sont décrits dans la section 1.3. À terme, nous souhaitons mettre en place une solution de synthèse hybride où les *embeddings* issus de réseaux de neurones permettent de définir automatiquement une distance entre nos unités de paroles, servant ainsi de coût de sélection, la recherche des meilleurs unités candidates s'effectuant alors par une recherche de plus proches voisins, éventuellement approximative.

1.2 Réseaux de neurones

Les réseaux de neurones sont une technique d'apprentissage automatique supervisé. Une utilisation alternative de cette technique est la production de représentations continues, souvent compactes, de données fournies en entrée. Après un rappel sur les réseaux de neurones, ce sont ces représentations, appelées *embeddings*, qui feront l'objet d'une étude particulière. Nous concluons enfin cette section par l'étude d'une technique qui permet d'améliorer les performances d'un DNN, l'apprentissage multi-tâches.

1.2.1 Généralités

Les réseaux de neurones reposent sur le modèle du perceptron. Un perceptron est un neurone possédant plusieurs entrées x_i auxquelles sont associées un coefficient w_i, une sortie s et une fonction d'activation f à laquelle est associée un seuil w_0. La sortie est prédite comme la somme des entrées pondérée par leur coefficient, à laquelle la fonction d'activation est appliquée en prenant en compte le décalage introduit par le seuil, soit $s = f(\sum_i w_i x_i - w_0)$.

Un apprentissage comparant les valeurs prédites par le réseau et les valeurs attendues permet de faire converger les coefficients w_i vers des valeurs permettant une prédiction plus ou moins précise de la sortie en fonction du jeu de données utilisé. On peut alors introduire la notion de couche : plusieurs perceptrons partageant les mêmes entrées et la même fonction d'activation mais possédant des coefficients et sorties distincts. On peut alors considérer une couche de perceptrons sous une représentation vectorielle dont chaque composante correspond à une sortie : $s_j = f(\mathbf{w_j}^T \mathbf{x} - w_{j_0})$ où $\mathbf{w_j}$ est le vecteur représentant les coefficients permettant de prédire la sortie j et $\mathbf{x}$ est le vecteur représentant les entrées de la couche.

Il est ensuite possible de construire des réseaux de neurones possédant plusieurs couches, on parle alors de DNN. Dans ces types de réseaux, les sorties d'une couche cachée correspondent aux entrées de la couche suivante. Une couche peut être entièrement connectée à la précédente, on parle alors de couche dense, ou partiellement connectée, on parle de couche convolutionnelle. Dans un cas comme dans l'autre le principe reste le même : les sorties correspondent à une combinaison des entrées. En général, la dimension des couches cachées [2] d'un DNN est plus petite que celle de ses entrées, chaque couche possède ainsi une représentation différente de l'entrée. Ce sont ces représentations alternatives que l'on appelle *embeddings*.

1.2.2 Embeddings

Calculer un *embedding* correspond à entraîner un DNN pour une tâche particulière dans le but d'utiliser non pas la sortie prédite par le réseau mais les valeurs en sortie de l'une des couches cachées. Les *embeddings* sont souvent utilisés afin d'encoder un contexte large à l'aide d'un vecteur de plus petite dimension. Par exemple, dans [9] les auteurs introduisent le modèle *skip-gram*. Représenté sur la figure 3, ce modèle est un réseau de neurones entraîné à prédire, à partir de la représentation vectorielle d'un mot en entrée, la représentation des mots l'entourant dans la phrase d'origine : ceux le précédant et ceux le suivant. Une fois entraîné, un tel modèle permet de contenir la représentation condensée des mots en entourant un autre. De manière inverse, on peut tenter de prédire un mot à partir de ceux environnants, ce qui s'appelle un *Continuous Bag of Words* (CBOW).

2. Couche d'un réseau située entre l'entrée et la sortie.

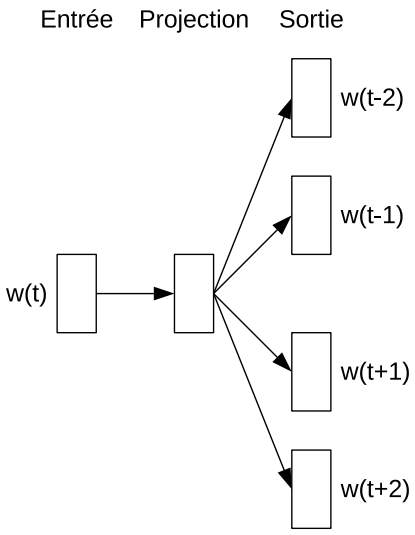

FIGURE 3 – Un *skip-gram* est un modèle tentant de prédire les mots (ici 2) qui précèdent et suivent le mot courant au sein d'un énoncé. (Source : [9])

Sous cette forme, on peut considérer que les *embeddings* sont une forme de projection d'un espace vectoriel de grande taille vers un espace vectoriel de taille inférieure. Dans [10], les auteurs mettent en avant un autre aspect intéressant des *embeddings* : ils permettent de conserver les similarités entre objets dans cet espace projeté et les projections de deux objets partagent les mêmes liens que les deux objets originaux. Par exemple, si l'on examine les représentations vectorielles issues de l'*embedding* de noms de différents pays par le *skip-gram*, on observe que ces représentations sont réparties selon un axe. La même expérience effectuée sur le nom de différentes capitales répartit ces dernières sur un même axe parallèle à celui des pays. On remarque de plus que le vecteur reliant l'*embedding* d'un pays à celui de sa capitale est quasi-égal à celui reliant l'*embedding* d'un autre pays à celui de sa propre capitale. Dans le même article, les auteurs proposent de définir des opérations sur les *embeddings* de mots en tant que vecteurs afin de répondre à des requêtes analogiques du type « Quel est le mot qui se rapproche de 'France' de la même manière que 'Madrid' est proche d' 'Espagne' ? » et montrent que les *embeddings* de mots issus de *skip-gram* permettent de répondre 'Paris'.

Les *embeddings* ne sont pas uniquement utilisés pour la représentation de mots mais aussi en traitement d'image ou pour améliorer la reconnaissance de phonèmes. Dans [11], la reconnaissance d'action appliquée à la vidéo nécessite l'apprentissage d'attributs spatio-temporels caractérisant la vidéo avant d'utiliser un classifieur. L'apprentissage d'*embeddings* permet ici d'apprendre des attributs sans nécessiter d'expertise quant au choix de ces attributs. Dans [12], l'auteur tente d'améliorer la reconnaissance de phonèmes grâce à l'utilisation d'*embeddings*. Ceux-ci sont issus d'un réseau entraîné à prédire la description acoustique d'un phonème à partir de son contexte. Les *embeddings* servent ici à condenser le contexte d'un phonème qui va servir d'entrée à un second DNN. Il est intéressant de noter que comme dans cet article, l'apprentissage d'*embeddings* peut être utilisé conjointement avec les autres techniques usuelles en construction de DNN, et notamment l'apprentissage multi-tâches.

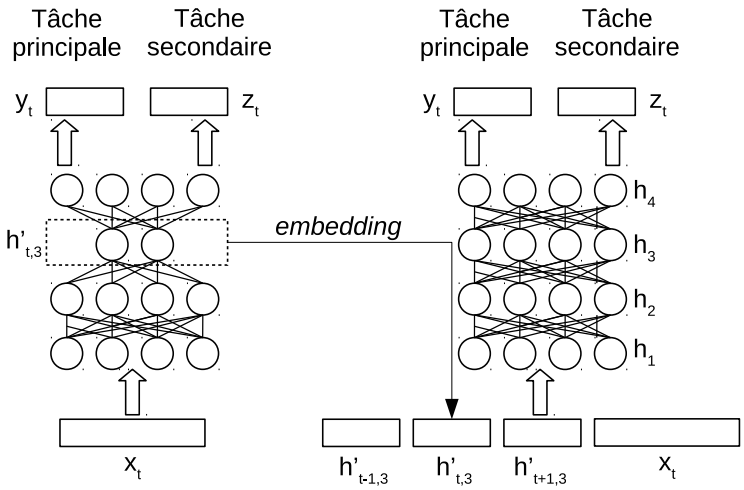

FIGURE 4 – Exemple d'un DNN liant multi-tâches et *embeddings* (Source : [13]). Un premier réseau multi-tâches est entraîné afin d'extraire des embeddings qui permettent de condenser le contexte d'un phonème et servent ensuite d'entrée à un second réseau multi-tâches.

1.2.3 Apprentissage multi-tâches

Tout réseau de neurone est entraîné dans un but particulier. L'idée de l'apprentissage multi-tâches est d'ajouter une tâche au réseau de neurones lors de l'apprentissage. On se rend alors compte que la précision des prédictions du DNN augmente généralement. En effet, ajouter une autre tâche permet au DNN de généraliser son analyse des données d'entrée en apprenant des caractéristiques liées à la seconde tâche. Il est intéressant de remarquer que différentes tâches secondaires n'apportent pas le même gain de performances.

Par exemple, dans [14], différentes tâches secondaires sont proposées afin d'améliorer la reconnaissance de phonèmes. Un DNN est entraîné comme modèle acoustique en tant que tâche principale et les différentes tâches secondaires testées sont les suivantes : prédire les états acoustiques (comparés avec les résultats d'un *k-means*) ou prédire le contexte de la trame en cours de traitement (trame précédente et suivante), un réseau sans tâche secondaire est entraîné à titre de comparaison. Pour chaque modèle, une mesure du taux d'erreur phonème est effectuée afin d'évaluer la qualité de leurs prédictions en termes de classification. Ces mesures montrent que les modèles multi-tâches ont tous de meilleurs résultats que le modèle simple.

De manière similaire dans le domaine de la synthèse de la parole, les auteurs de [13] montre que l'apprentissage multi-tâches permet d'apprendre un modèle acoustique plus performant qu'un modèle fondé sur un HMM ou un DNN simple. Deux réseaux multi-tâches sont entraînés (*cf.* figure 4). Leur tâche principale est de prédire les coefficients acoustiques des descripteurs de phonèmes en entrée et la tâche secondaire varie selon les expériences. Cette étude observe aussi les améliorations apportées par l'utilisation ou non dans le deuxième réseau d'un contexte sous forme d'*embeddings*, ces derniers étant issus du premier réseau. Des mesures objectives telles que l'erreur quadratique moyenne sur les coefficients acoustiques prédits montrent que l'utilisation d'apprentissage multi-tâches améliore le taux d'erreur et que cette amélioration est renforcée par l'utilisation de contexte sous forme d'*embeddings* . En revanche, les auteurs n'ont pas réussi à

montrer d'amélioration significative lors des tests d'écoute.

Au final, grâce à un DNN entraîné pour calculer des *embeddings*, on peut obtenir une projection d'un ensemble d'unités de parole dans un même espace vectoriel. Si l'utilisation d'apprentissage multi-tâches permet d'augmenter la performance du réseau, on peut espérer que les *embeddings* issus de ce réseau soient aussi de meilleure qualité. Dans l'optique d'une synthèse par sélection d'unités, il se pose alors la question de la méthode pour sélectionner une unité voulue dans ce nouvel espace projeté. Pour cela, les techniques d'indexation sont une piste de solution, et même plus particulièrement les techniques de recherche de plus proches voisins en grande dimension. C'est précisément l'objet de notre prochaine section.

1.3 Recherche de plus proches voisins dans un espace de grande dimension

La recherche de plus proches voisins consiste à rechercher une donnée particulière dans une base. Si celle-ci existe, on doit être capable de la trouver, sinon on doit être capable de trouver la ou les donnée(s) s'en rapprochant le plus. Ce point est intéressant vis-à-vis de la sélection d'unités où cette recherche permettrait de trouver les unités semblables à celles demandées par l'analyse linguistique. Le fait de considérer plusieurs voisins permet alors de choisir parmi toutes les combinaisons possibles à l'échelle de la phrase la où les unités qui se concatènent le mieux.

Si une recherche exhaustive de plus proches voisins fonctionne, celle-ci est loin d'être optimale sur un plan algorithmique et le recours à des techniques d'indexation s'avère plus efficace. Pour obtenir une voix expressive en synthèse de la parole, la quantité de données stockées et la grande dimension de leurs descripteurs poussent à explorer de nouvelles solutions de recherche, notamment dans le domaine de la recherche d'informations. La plupart des solutions existantes peuvent être réparties en deux catégories : les solutions exactes et les solutions approximatives qui n'assurent pas que les points retournés fassent partis des plus proches voisins du point cherché. Une autre distinction possible consiste à opposer les arbres de partitions et les techniques à base de hachage.

1.3.1 Arbres de partitions : kd-tree

Les kd-tree permettent de partitionner spatialement les points d'un espace vectoriel en s'inspirant d'un arbre de recherche binaire [15]. Dans le cas d'un arbre de recherche binaire, on sépare un ensemble de valeurs de dimension 1 en construisant un nœud dont la valeur est la médiane de l'ensemble, son fils gauche correspond à l'ensemble des valeurs inférieures à la médiane et le fils droit correspond aux autres valeurs. La construction est effectuée de manière récursive sur les fils gauche et droit avec l'ensemble de valeurs associées restantes. On arrête la récursivité quand le nombre de valeurs associées à un nœud est suffisamment petit. Si une valeur est seule, on la stocke telle quelle, sinon on stocke l'ensemble des valeurs sous la forme d'une liste chaînée. L'ensemble des valeurs est donc stocké dans les feuilles, aucune ne se trouvent dans un nœud interne.

Les kd-tree, quant à eux, étendent ce principe à des ensembles de dimension k. Il faut alors pour chaque nœud choisir selon quelle dimension partitionner l'espace. En général, on choisit la dimension selon laquelle la variance est maximale. De manière similaire aux arbres de recherche binaires, on divise l'espace en séparant les points au niveau de la médiane de la composante. On recommence récursivement sur les deux sous-ensembles jusqu'à avoir des ensembles suffisamment petits. La recherche de plus proches voisins dans un kd-tree revient alors à comparer la i-ème composante du point recherché avec la valeur du nœud où i est la composante discriminante choisie

lors de la construction. Si la composante est inférieure, on examine récursivement le fils gauche, sinon on examine récursivement le fils droit. Les résultats de la recherche correspondent aux points trouvés au niveau des feuilles.

Malheureusement, les kd-tree ont une complexité linéaire en temps en fonction de la dimension et du nombre de données pour la recherche. Ils ne sont donc pas adaptés aux espaces de grande dimension. Il convient donc de les adapter, quitte à n'obtenir qu'une solution approximative. Nous verrons une adaptation dans la section 1.3.3, mais commençons par examiner les techniques fondées sur le hachage.

1.3.2 Locality Sensitive Hashing (LSH)

La méthode LSH se base sur le principe de projections depuis l'espace de recherche vers un espace de dimension réduite sous l'hypothèse que deux points proches dans l'espace original ont une forte probabilité de l'être aussi dans l'espace projeté et, de manière similaire, deux points éloignés dans l'espace d'origine ont une forte probabilité d'être éloignés dans l'espace de projection. Plus précisément, selon [16], on souhaite avoir une famille de fonctions de hachages H (chaque fonction de hachage définit un espace projeté différent) telles que pour $h \in H$, on puisse trouver un rayon R, un coefficient c et deux probabilités constantes P_1 et P_2 telles que pour tous points p et q de l'espace de recherche on ait les deux propriétés suivantes :

$$||p - q|| \leq R \Rightarrow Pr(h(q) = h(p)) \geq P_1$$
$$||p - q|| \geq cR \Rightarrow Pr(h(q) = h(p)) \leq P_2$$

L'algorithme LSH effectue le hachage sur la concaténation de L fonctions de hachages de H : $g_j(q) = (h_{1,j}(q), ..., h_{k,j}(q)), j \in 1, ..., L$ où k est un paramètre et q un point de l'espace vectoriel. Alors, on construit L tables de hachages à partir de ces fonctions et on y répartit l'ensemble des points du jeu de données. La recherche de plus proches voisins d'un point q se fait en examinant les cases des L tables ayant la même valeur de hachage que q. On examine la distance entre les points retournés par cette méthode et le point cherché. Si la distance est satisfaisante, on considère que le point trouvé fait partie des plus proches voisins.

L'algorithme LSH présente l'avantage d'une recherche rapide : la complexité temporelle est constante, il n'y a collision qu'avec une seule case pour chacune des L table de hachage, on parcourt donc entre 1 et L cases en fonction du nombre de plus proches voisins désirés et le nombre de points contenus dans les cases avec lesquelles il y a collision. En revanche, l'algorithme peut nécessiter beaucoup de mémoire car possède une complexité spatiale en $O(nL)$ où n représente le nombre de données dans la base. Autre défaut, LSH effectue une recherche approximative et de nombreux paramètres sont à régler et influent énormément sur les performances de l'algorithme. De plus, il faut être capable de définir une distance entre les points de l'espace vectoriel.

1.3.3 Autres méthodes de recherche approximative

La méthode fondée sur les kd-tree vue précédemment permettait d'effectuer une recherche exacte de plus proches voisins. Cependant, quand la dimension de l'espace de recherche et le nombre de données augmente, le temps d'exécution d'un tel algorithme devient vite incompatible avec une application interactive telle que la synthèse de la parole. Dans [17], les auteurs remarquent que se limiter à une recherche approximative des plus proches voisins permet de retourner 90% de

plus proches voisins corrects en obtenant des résultats au moins deux fois plus rapidement. Les auteurs proposent deux méthodes approximatives fondées sur la partition de l'espace de recherche : la construction de kd-tree aléatoires et la construction d'arbres k-means hiérarchiques.

Le premier algorithme consiste à construire plusieurs kd-tree de manière aléatoire. Lors du choix de la composante qui permet de séparer l'espace, plutôt que de choisir la dimension ayant la plus grande variance, la composante est choisie aléatoirement parmi les 5 dimensions présentant la plus grande variance. On obtient alors une forêt de kd-tree. Les différents arbres de la forêt sont explorés parallèlement avec une file de priorité partagée. La file est triée par distance croissante entre la frontière représentée par le nœud en cours d'analyse et le point cherché. Ainsi, la recherche examinera en priorité les feuilles les plus proches du point recherché. Enfin, dès qu'un point est examiné par la recherche dans un arbre, il est marqué comme tel afin de ne pas être à nouveau examiné dans les autres arbres. L'algorithme se termine quand un certain nombre de feuilles ont été parcourues et renvoie les voisins trouvés jusqu'à présent.

Le second algorithme consiste à appliquer un k-means sur l'ensemble du jeu de données, puis continuer récursivement à appliquer un k-means sur chaque cluster obtenu dans l'étape précédente jusqu'à obtenir un certain nombre de points par cluster. On obtient alors une partition de l'espace sous la forme d'un arbre dont les fils de chaque nœud sont les clusters obtenus par application du k-means. Lors de la recherche de plus proches voisins, on parcourt l'arbre en prenant à chaque étape la branche dont le centre de cluster correspondant est le plus proche du point cherché.

Au final, les techniques de recherche de plus proches voisins vues ici imposent de faire un compromis. Si les kd-tree peuvent proposer une solution efficace, cette méthode est sujette au fléau de la dimension. Certaines adaptations permettent aux kd-tree d'être plus efficaces dans un espace de haute dimension. Dans la catégorie des recherches approximatives, la solution LSH semble aussi être efficace mais nécessite de régler de nombreux paramètres. Il est intéressant de tester ces différentes solutions sur son propre jeu de données, car la performance de ces algorithmes semblent en être dépendante.

2 Construction et étude d'une méthode hybride

Ce stage s'effectue dans le contexte des méthodes hybrides de synthèse de parole, en particulier, nous tentons de mettre en place une méthode de synthèse par sélection d'unités guidée par réseaux de neurones. L'idée consiste à apprendre un coût de sélection de manière automatique, sans expertise linguistique. Pour cela, nous proposons d'utiliser la propriété d'*embeddings* des réseaux de neurones. La méthode hybride (schématisée sur la figure 5) nécessite tout d'abord d'entraîner un réseau de neurones pour une tâche particulière (on se concentre ici sur la prédiction soit de descripteurs linguistiques, soit de descripteurs acoustiques, ou bien des deux descripteurs à la fois) et d'utiliser la représentation issue d'une couche cachée pour définir un espace de projection où l'on peut définir implicitement une distance entre nos différentes unités de parole. À terme, après avoir projeté l'ensemble des unités de notre base de données dans l'espace d'*embeddings*, on peut effectuer notre sélection d'unités en projetant l'unité recherchée et en effectuant une recherche de plus proches voisins : le coût de sélection utilisé ici correspond à la distance euclidienne définie implicitement dans l'espace de projection. Il est envisageable d'utiliser une méthode de recherche de plus proches voisins approximative pour accélérer la recherche des unités à concaténer, au prix de faux positifs pouvant éventuellement diminuer la qualité de la parole synthétisée. Le nombre de données utilisées

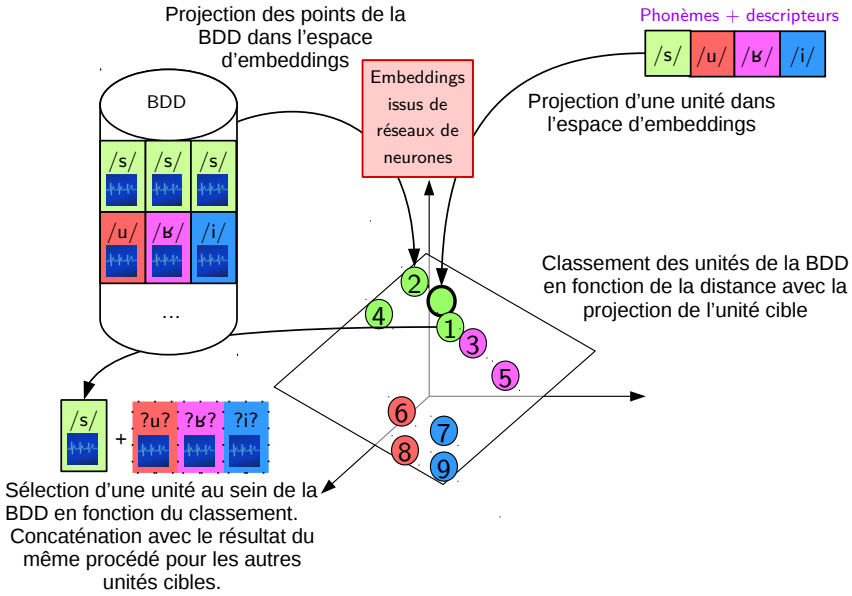

FIGURE 5 – La méthode proposée ici consiste à projeter l'ensemble des unités de notre base de données dans un espace métrique via les *embeddings* issus d'un réseau de neurones. La synthèse s'effectue alors en concaténant la première unité à l'issue d'une recherche de plus proche voisins de l'unité recherchée pour chaque unité souhaitée.

lors du stage ne justifiant pas l'utilisation de méthodes approximatives, nous avons utilisé une méthode de recherche de plus proches voisins classique.

Dans la suite de ce rapport, nous commençons par présenter le jeu de données utilisé lors du stage, qui contient à la fois des descripteurs linguistiques et des descripteurs acoustiques. Ce jeu de données nous permettra ensuite d'entraîner différents types de réseaux de neurones. Nous proposons ici d'examiner des auto-encodeurs, des modèles acoustiques ainsi que des modèles multi-tâches. Nous chercherons une architecture optimale pour chacun de ces modèles en évaluant leur précision indépendamment de la qualité de leurs *embeddings*. Une fois une architecture optimale trouvée, nous pourrons extraire les *embeddings* associés et nous intéresser à l'évaluation de leur qualité vis-à-vis de leur utilisation pour définir un coût de sélection. Cette évaluation est effectuée à partir de mesures se basant sur la conservation des distances entre points dans l'espace d'origine lors de la projection dans l'espace d'*embeddings* et sur le rang à l'issue d'une recherche de plus proches voisins.

2.1 Données

Le but de cette section est de présenter le jeu de données utilisé lors du stage. Il s'agit de présenter de présenter les attributs considérés comme utiles pour l'apprentissage d'*embeddings* avant d'étudier quelques propriétés statistiques de ces attributs.

Attributs	Type	Dimension
Classe de phonème	Booléen	36
Étiquette morpho-syntaxique	Booléen	118
Position début/fin dans proposition	Booléen	2
Structure syllabe (Attaque/Coda)	Booléen	4
Position mot/syllabe par rapport aux NSS	Réel	4
Nombre mots/syllabes dans proposition courante/précédente/suivante	Réel	6
Position syllabe dans mot et syllabe/mot dans proposition	Réel	10
Position unité dans syllabe, taille syllabe	Réel	3
Structure de l'énoncé	Réel	5
Position mot dans proposition	Réel	2
Caractère ascendant/descendant phonème/syllabe courant	Booléen	4
Descripteurs articulatoires	Booléen	48
Consonne/Voyelle	Booléen	1
Coefficients MFCC	Réel	39 par trames
Durée phonème	Réel	1
$\log(f_0)$	Réel	1

TABLE 1 – Ensemble des attributs constituant le vecteur de description linguistique (partie haute) et le vecteur de description acoustique (partie basse).

2.1.1 Attributs

Pour effectuer une synthèse par sélection d'unités, il faut tout d'abord disposer d'une base de données contenant des segments de parole. Dans notre cas, il s'agit d'un *audiobook* correspondant au roman « À la recherche du temps perdu - Albertine disparue » de Marcel Proust et lu par Denis Podalydès. Une première étape d'alignement est effectuée afin de faire correspondre le texte original et le signal audio résultant de la lecture. Cette étape est suivie d'une découpe des signaux audios à l'échelle du phonème et d'une analyse linguistique et prosodique afin d'obtenir différentes informations pour chaque unité de notre jeu de données. Ces étapes sortant du cadre du stage, nous ne développerons pas leur déroulement mais nous en examinerons le résultat. Pour chaque unité, les informations suivantes sont extraites et sont rappelées dans le tableau 1.

- La classe de phonème. Il existe en tout 36 phonèmes dans la langue française, la classe est encodée sous la forme d'un vecteur *one-hot* de taille 36.
- L'étiquette morpho-syntaxique indiquant les informations grammaticales du mot contenant l'unité (nom, verbe, adjectif, adverbe, etc.). De même, cette information est encodée sous la forme d'un vecteur *one-hot*, de taille 118.
- La position du mot contenant l'unité dans la proposition : est-il au début, à la fin ?
- La structure de la syllabe dans laquelle se situe l'unité. Chaque syllabe se découpe en deux éléments, l'attaque (éventuellement absente) et la rime, elle-même composée d'un noyau et d'une coda (éventuellement absente). On extrait donc la présence ou l'absence d'une attaque et d'une coda, ainsi que la présence ou l'absence de l'unité dans l'attaque et la coda.
- La position de la syllabe et du mot par rapport aux NSS (*Non-Speech Sounds*), sons ne faisant pas partie du discours, comme par exemple le silence, un sifflement ou une hésitation. On compte le nombre de syllabes et de mots entre l'unité et les NSS précédents et suivants.
- La structure de la proposition à laquelle l'unité appartient ainsi que celle de la proposition

précédente et suivante : le nombre de syllabes et de mots par proposition.
— La position de la syllabe contenant l'unité dans le mot contenant l'unité. Cette position est mesurée dans les deux sens, entre le début du mot et la syllabe, puis entre la fin du mot et la syllabe. De manière similaire, on mesure la position de la syllabe dans la proposition et la position du mot dans la proposition.
— La position de l'unité dans la syllabe la contenant, la taille de la syllabe contenant l'unité.
— La structure de l'énoncé : la position de la proposition dans l'énoncé, le nombre de syllabes, de mots et de propositions dans l'énoncé.
— La position du mot dans la proposition.
— La nature de la syllabe courante et du phonème courant : l'intonation est-elle montante ou descendante ?
— Les descripteurs articulatoires pour le phonème en cours : position de l'articulation, forme de la bouche, etc.
— L'unité en cours est-elle dans une consonne ou une voyelle ?

Toutes les valeurs entières et réelles décrites ci-dessus sont normalisées pour prendre des valeurs entre 0 et 1 en divisant chacune par leur maximum sur l'ensemble du jeu de données. Par exemple, le nombre de syllabes dans un énoncé est divisé par le nombre de syllabes dans l'énoncé en contenant le plus. On obtient au final un ensemble de 243 attributs constituants le vecteur de description linguistique de nos unités. Ces attributs sont les seuls qui seront présents lors de l'utilisation du moteur de synthèse et à partir desquels on cherche à définir un coût de sélection automatiquement grâce aux *embeddings* issus de réseaux de neurones. Cependant, rien ne nous empêche lors de l'entraînement des réseaux de tenter de prédire des données qui ne seront pas présentent lors de l'utilisation du moteur de synthèse. Des données supplémentaires sont donc extraites, cette fois relative à la description acoustique de nos unités. Les données suivantes sont rappelées dans le tableau 1 :

— Un vecteur de coefficients cepstraux (MFCC, *Mel-Frequency Cepstral Coefficients*) permettant de décrire le signal. Il s'obtient en effectuant une transformation de Fourier puis une série d'opérations de traitement du signal [18]. On se limite pour chaque unité, à intervalle régulier de 10 ms, à extraire 13 coefficients statiques, leur dérivée première et seconde. Au final, on obtient un vecteur MFCC de dimension 39 pour chaque trame.
— La durée du phonème.
— Le logarithme de la fréquence fondamentale f_0 utilisée lors de la prononciation de l'unité.

L'ensemble de ces attributs forment le vecteur de description acoustique. Il permet d'obtenir une représentation de la manière dont chaque unité est produite.

2.1.2 Statistiques

Le jeu de donnée décrit dans la section précédente correspond à une dizaine d'heures de paroles qui a été divisé en 3336 énoncés pour un total de 390 000 unités à l'échelle du phonème. On le divise en un sous-ensemble d'entraînement de 3005 énoncés (350 000 unités) et des sous-ensembles de validation et de test contenant respectivement 165 et 166 énoncés (20 000 unités). Dans un premier temps, nous explorons le jeu de données en regardant la proportion de chaque phonème au sein des différents sous-ensemble. Ces proportions sont représentées sur la figure 6.

Il est intéressant de remarquer que la proportion de chaque phonème dans les trois jeux de données est très similaire, on ne risque donc pas de souffrir de disparités en passant de l'entraînement à l'évaluation. Cependant, certains phonèmes comme /ɲ/ (a**gn**eau) et /ŋ/ (campi**ng**) sont sous-

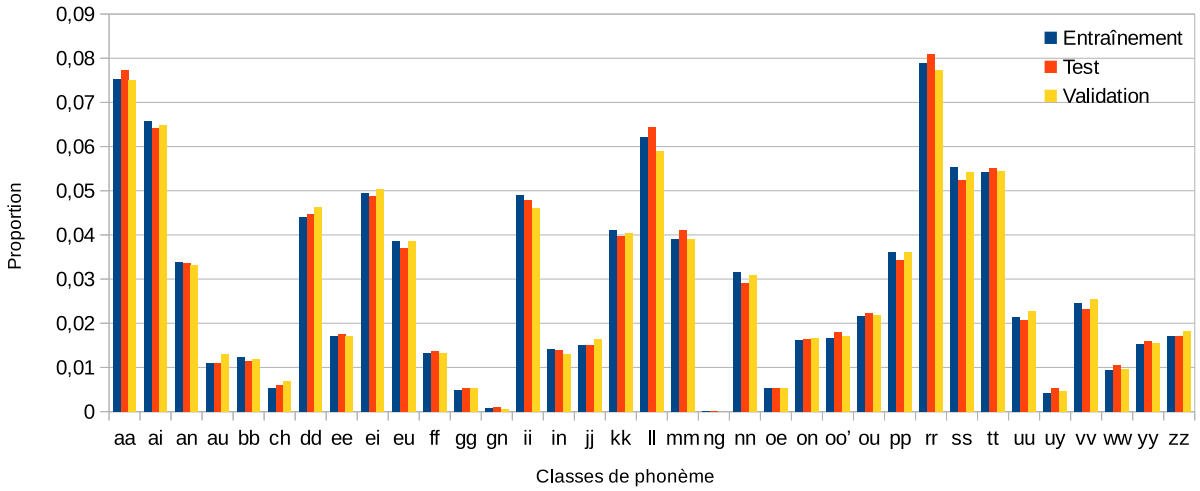

FIGURE 6 – Proportion de chaque phonème par jeu de données

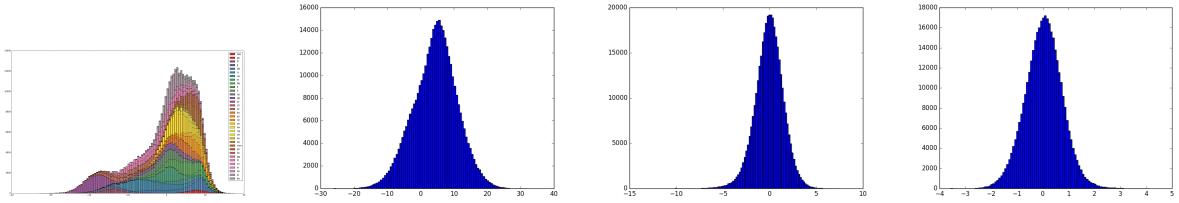

FIGURE 7 – De gauche à droite, les histogrammes du premier coefficient MFCC statique, du 3ème coefficient MFCC statique, des dérivées première et seconde du 3ème coefficient MFCC statique. La première figure est reproduite en annexe.

représentés par rapport à des phonèmes comme /a/ (ch**a**t) ou /ʁ/ (sou**r**is). Les proportions obtenues dans notre jeu de données reflètent les proportions de la langue française de manière générale mais, dans la perspective d'appliquer une méthode d'apprentissage automatique telle que des réseaux de neurones, il faut garder à l'esprit que les disparités peuvent se traduire par un modèle privilégiant les unités majoritaires et n'ayant rien appris sur les unités minoritaires.

Attachons-nous maintenant à examiner les descripteurs acoustiques en observant la population pour chaque dimension, les histogrammes étant visibles en figure 7. Il s'avère que, le premier coefficient MFCC statique mis à part, les coefficients statiques, leur dérivée première et seconde suivent tous une loi normale dont les paramètres varient pour chaque coefficient et, de manière logique, l'amplitude des valeurs des coefficients statiques est supérieure à celle des dérivées première, elle-même supérieure à celle des dérivées seconde. Pour revenir sur le premier coefficient statique, sa distribution ne suit pas une loi normale car présente un nombre de valeurs non négligeable inférieures à la moyenne. Cette répartition peut être expliquée par l'interprétation de ce coefficient qui est associé à l'énergie. Les phonèmes de basse énergie comme /s/ (**s**ale) ou /ʃ/ (**ch**at) sont moins présents que ceux à haute énergie comme /a/ (pl**a**t) ou /ɛ/ (col**è**re) comme on l'a vu précédemment. Dans la suite, on considère uniquement les coefficients MFCC associées aux 3 trames centrales, là

ou le signal est le plus stable.

2.2 Modèles étudiés

L'objectif de ce stage est de définir une distance entre descripteurs linguistiques de manière automatique grâce aux *embeddings* issus de réseaux de neurones. Cette section s'attache à présenter les différents modèles envisagés pour l'extraction d'*embeddings*, à commencer par l'architecture partagée par les différents modèles étudiés : un auto-encodeur linguistique, un modèle acoustique et un modèle multi-tâches. Pour chaque modèle, une architecture optimale sera recherchée en évaluant chacun d'eux sur la précision de ses prédictions en présupposant que la qualité des embeddings d'un réseau de neurones est liée à la qualité de ses prédictions. L'évaluation à proprement parler des embeddings fait l'objet de la section 3.

2.2.1 Architecture générale

Lorsque l'on utilise des réseaux de neurones, de nombreux hyper-paramètres nécessitent d'être réglés. Ils peuvent être divisés en deux catégories : ceux relatifs à l'architecture du réseau et ceux relatifs à son entraînement. Le but de cette section est de mettre en avant les hyper-paramètres communs à tous les modèles envisagés lors de ce stage.

Du point de vue de l'architecture, tous les réseaux travaillent sur la même entrée correspondant aux descripteurs linguistiques. L'architecture générale possède donc une entrée de dimension 243. Lorsque l'on cherche l'architecture optimale d'un réseau, nous faisons varier le nombre de couches cachées et leur dimension de la manière suivante :

— Une couche cachée possède une dimension variant entre 1 et 64 et, dans le cas où il y a plus d'une couche cachée, seule la dimension de la couche cachée la plus profonde varie tandis que celle des autres est fixée à 64. Nous procédons de la sorte afin de faire varier le nombre de couches cachées tout en gardant une couche de dimension réduite qui permettra d'obtenir des *embeddings* de dimension variable. En effet, nous avons décidés de toujours extraire les embeddings à partir de la couche cachée la plus profonde.

— La fonction d'activation de toutes les couches cachées est une sigmoïde qui peut s'exprimer de la manière suivante en reprenant les notations de la section 1.2.1 :

$$s_i = \frac{1}{(1 + e^{-(\mathbf{w_i}^T \mathbf{x} - w_{i_0})})}.$$

Pour la couche de sortie de réseau de neurones, puisque la nature des données à prédire varie en fonction de la tâche associée au réseau, le nombre de neurones en sortie est variable. Cependant, la sortie du réseau utilise toujours une fonction d'activation linéaire (équation 1) ou *softmax* (équation 2).

$$s_i = \mathbf{w_i}^T \mathbf{x} - w_{i_0} \tag{1}$$

$$s_i = \frac{e^{\mathbf{w_i}^T \mathbf{x} - w_{i_0}}}{\sum_{j=1}^{K} e^{\mathbf{w_j}^T \mathbf{x} - w_{j_0}}} \tag{2}$$

Une fonction d'activation linéaire joue simplement un rôle d'agrégation sans ajouter de non-linearité supplémentaire, elle est appropriée dans un problème de régression. Dans le cas d'une fonction *softmax*, la sortie de chaque neurone est comprise entre 0 et 1, leur somme vaut 1. La sortie de

chaque neurone est donc assimilable à une probabilité. Cette fonction d'activation est appropriée dans un problème de classification.

Du point de vue de l'entraînement, l'optimisation de la fonction de coût est effectuée à l'aide de l'algorithme RMSPROP [19]. L'entraînement s'effectue sur 1000 *epochs* au maximum, l'apprentissage étant arrêté automatiquement si la fonction de coût appliquée au jeu de validation ne montre pas d'amélioration sur 30 epochs successives. Pour accélérer l'apprentissage, celui-ci est effectué par *mini-batch*[3] de taille 512. Il ne reste alors plus qu'à définir la fonction de coût à minimiser. En fonction du modèle, on utilise soit l'erreur quadratique moyenne (*Mean Squared Error*, MSE), soit l'entropie croisée. Pour p les données réelles et q les prédictions en sortie du réseau de neurones, l'entropie croisée est définie par

$$H(p,q) = \sum_x p(x) log(q(x)).$$

On utilise plutôt la MSE dans des problèmes de régression et l'entropie croisée dans des problèmes de classification. Les réseaux de neurones que nous allons présenter dans les sections suivantes sont entraînés à l'aide du *framework* Keras en utilisant le moteur TensorFlow [20].

2.2.2 Auto-encodeur linguistique

Si pour l'entraînement de nos réseaux de neurones on se limite aux données présentes lors de la synthèse de parole, c'est-à-dire les descripteurs linguistiques, le modèle le plus simple à envisager correspond à un auto-encodeur. Un auto-encodeur est un réseau de neurones dont la tâche consiste à prédire les données qui lui ont été fournies en entrée. De manière générale, un auto-encodeur possède toujours une couche cachée de dimension inférieure à celle des autres forçant le réseau à apprendre une représentation condensée des données d'entrées. Cette couche cachée peut être considérée comme la séparation du réseau en deux blocs fonctionnels : un encodeur suivi d'un décodeur. L'avantage d'un tel réseau est que la qualité de l'encodage (et donc de la projection dans l'espace des *embeddings*) est directement liée à la précision du réseau lui-même, on peut donc considérer que l'architecture optimale du point de vue de la précision des prédictions donnera aussi le meilleur espace de projection.

L'architecture optimale de nos auto-encodeurs linguistiques est recherchée en minimisant la MSE entre les descripteurs linguistiques réels et les valeurs prédites par des réseaux de neurones à 1 ou 2 couches cachées, de sortie linéaire de même dimension que l'entrée, égale à 243. On s'attend pour le premier modèle à obtenir une MSE qui diminue en fonction de la dimension de la couche cachée, car l'augmentation du nombre de paramètres à disposition devrait permettre d'améliorer la précision du modèle. Pour le réseau à deux couches cachées, en fonction de la dimension de la seconde couche cachée, on espère trouver une MSE plus faible, ou tout du moins qui décroît plus vite, due à un plus grand nombre de paramètres que pour le premier modèle.

La figure 8 présente la MSE obtenue sur le jeu de test pour l'auto-encodeur à 1 ou 2 couches cachées en faisant varier la dimension de la couche cachée la plus profonde entre 1 et 64. On peut commencer par remarquer que pour tous les modèles, même lorsque la couche cachée la plus profonde est de dimension 1, la MSE est inférieure à 0,035. Si cette valeur peut paraître extrêmement faible, elle est à modérer par rapport aux valeurs manipulées : 243 valeurs comprises entre 0 et 1

3. La modification des poids du réseau est effectuée par paquet en calculant la moyenne des résultats de la fonction de coût sur le paquet plutôt qu'individuellement sur la totalité du jeu de données.

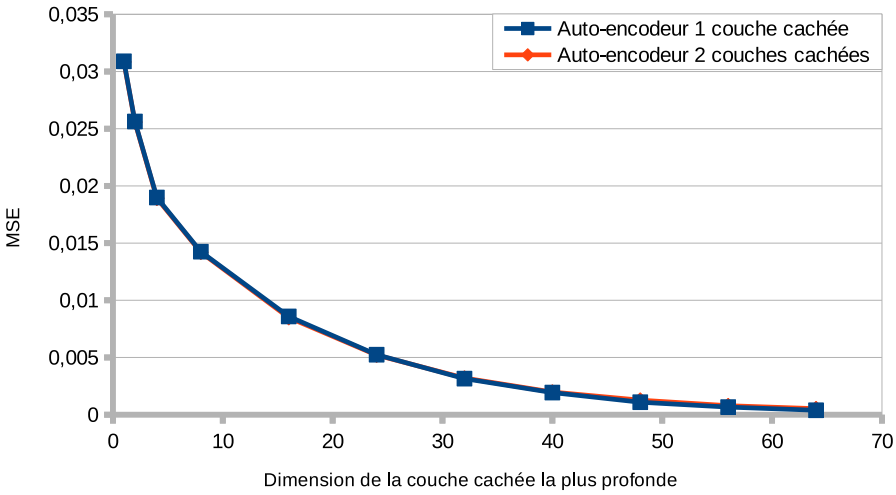

FIGURE 8 – MSE pour un auto-encodeur à 1 ou 2 couches cachées en fonction de la dimension de la couche la plus profonde.

dont la moyenne sur le jeu de données entier vaut 0,074. Ensuite, pour le premier modèle, on obtient les résultats attendus : une MSE qui diminue en fonction de la dimension de la couche cachée, de manière exponentielle pour tendre vers zéro. En revanche, on peine à distinguer le modèle possédant deux couches cachées de celui n'en possédant qu'une (le peu de différence n'étant pas significatif). Dans le cas de l'auto-encodeur à deux couches cachées, lorsque la dimension de la seconde couche cachée est inférieure à celle de la première, c'est-à-dire à 64, on pourrait penser que l'équivalence des MSE est due à une limitation par la seconde couche cachée. En revanche, lorsque la dimension des deux couches est égale, la potentielle limitation ne devrait plus exister, nous espérions donc avoir une amélioration de la MSE, ce qui n'est pas le cas ici. L'absence d'amélioration peut être due à deux effets :

— Augmenter le nombre de paramètres n'a pas d'effet car ceux de la première couche suffisent à modéliser notre phénomène, ou tout du moins on ne dispose pas d'assez de données pour tirer parti de plus de paramètres. On se trouve alors dans une situation de sur-apprentissage.

— Ajouter une seconde couche cachée dont la fonction d'activation est non-linéaire contribue à augmenter la distorsion de nos données, rendant plus difficile la tâche de décodage.

Les deux points précédents seraient probablement plus marqués si on augmentait le nombre de couches cachées de notre auto-encodeur, et alors l'augmentation de la MSE pour les plus grandes dimensions deviendrait probablement significative.

Nous avons vu que le second modèle est plus complexe que le premier sans apporter d'amélioration. C'est donc l'auto-encodeur à une couche cachée qui sera utilisé dans la suite. A priori, les *embeddings* issus de cet auto-encodeur sont une forme condensée des descripteurs linguistiques qui ne reflètent en rien le caractère acoustique de nos unités, caractère qui pourrait pourtant améliorer la qualité de la synthèse de parole finale.

2.2.3 Modèle acoustique

On profite désormais de toutes les données à notre disposition pour entraîner un réseau de neurones, y compris des données non présentes lors de l'utilisation du réseau pour la synthèse de parole. En particulier, dans cette section, on utilise les descripteurs acoustiques de nos unités : on tente de mettre en place des modèles acoustiques, des modèles dont la tâche correspond à prédire un vecteur de description acoustique à partir d'un vecteur de description linguistique. On espère que, comparés à ceux obtenus à partir d'auto-encodeurs linguistiques, les espaces projetés issus d'un modèle acoustique pourront refléter à la fois les attributs linguistiques et acoustiques de nos unités.

En cherchant l'architecture optimale d'un tel modèle, nous avons remarqué qu'il était compliqué de prédire la totalité des descripteurs acoustiques à partir des descripteurs linguistiques, en particulier les attributs dynamiques des MFCC, la durée des phonèmes et le logarithme de la fréquence fondamentale. Cela peut s'expliquer par l'étude statistique effectuée en section 2.1.2 : le réseau semble prioriser l'apprentissage des coefficients MFCC statiques dont l'amplitude est plus importante. Dans la suite, on se concentre sur la prédiction des coefficients MFCC statiques, c'est-à-dire les 13 premiers coefficients des vecteurs MFCC. La recherche de l'architecture optimale d'un modèle acoustique est effectuée en comparant 3 modèles comprenant respectivement 1, 2 et 3 couches cachées. Dans les 3 cas, la sortie est une couche linéaire de dimension 39 (13 coefficients statiques pour chacune des trois trames centrales). On évalue cette fois les modèles en mesurant le coefficient de corrélation R^2 de Pearson. Ce coefficient est un indicateur de la qualité du modèle : une valeur de 1 indique que les valeurs prédites correspondent parfaitement aux données réelles, une valeur de 0 indique que le modèle prédit la moyenne, une valeur négative indique que les prédictions sont moins précises que la moyenne. Nous utilisons ici et dans la suite la mesure de R^2 plutôt que la MSE dans le cas d'une tâche acoustique car l'erreur moyenne des prédictions ne tend pas vers 0 pour ces modèles. R^2 est alors plus simple à interpréter et comparer dans le cas d'une tâche acoustique. Pour chaque modèle, on s'attend à voir le coefficient R^2 augmenter en fonction de la dimension de la couche la plus profonde et qu'un modèle ayant plus de couches cachées soit plus performant qu'un modèle en possédant un nombre inférieur.

En observant les résultats obtenus sur la figure 9, on remarque que, en effet, tous les modèles ont bien un coefficient R^2 croissant avec la dimension des couches cachées, pour tendre vers une valeur de 0,37. De plus, le modèle à deux couches cachées est visiblement plus performant que celui à une seule couche cachée, bien que le passage de 2 à 3 couches cachées n'apporte pas réellement de précision supplémentaire : on peut supposer qu'il suffirait d'ajouter une ou deux couches cachées supplémentaires pour voir apparaître le phénomène de sur-apprentissage. Au vu de ces résultats, dans la suite, on considérera que le modèle acoustique par défaut correspond au modèle à deux couches cachées. On peut cependant s'interroger sur le sens de la valeur de 0,37 pour un coefficient de corrélation. La valeur étant strictement positive, cela indique que le modèle à au moins appris à prédire la moyenne de chaque composante. Cependant, cette valeur paraît faible et, puisque l'on a trouvé une architecture optimale pour cette tâche, on tente d'améliorer la prédiction de nos réseaux en transformant le problème de régression en un problème de classification.

2.2.4 Modèle acoustique discret

La première manière de transformer la prédiction des attributs acoustiques en un problème de classification consiste à discrétiser l'espace acoustique. Chacun des coefficients MFCC statique

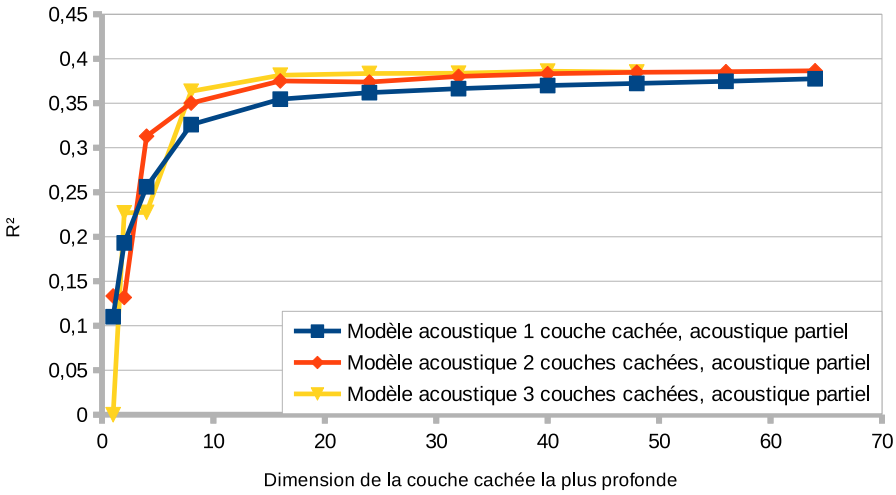

FIGURE 9 – Coefficient de Pearson pour un modèle acoustique à 1, 2 ou 3 couches cachées, en fonction de la dimension de la couche cachée la plus profonde.

prend des valeurs entre -30 et 30, si l'on définit un histogramme avec des intervalles de classe de 10, on obtient 6 classes différentes. On peut alors associer à chaque coefficient MFCC statique un vecteur *one-hot* encodant la catégorie à laquelle il appartient. On entraîne alors un réseau de neurones dont le rôle est de prédire, pour chaque coefficient MFCC statique, la probabilité d'appartenir à chacune des 6 catégories. Concrètement, cela se traduit par 39 groupes de 6 sorties, chaque groupe étant indépendant des autres et représenté par une couche *softmax*. Lors de l'entraînement du réseau, la fonction de coût à optimiser pour chaque sortie est la somme de l'entropie croisée pour chaque sortie et on peut évaluer nos modèles en mesurant la moyenne de la précision des prédictions pour chacun des 39 coefficients. La recherche d'une architecture optimale pour ce problème montre qu'avoir plus d'une couche cachée améliore peu la précision du réseau, on s'intéresse donc plutôt à l'effet du nombre de classes sur la précision. Puisqu'il n'est pas possible de comparer directement la performance d'un modèle acoustique classique avec un modèle acoustique discret, on effectue la comparaison avec un modèle prédisant une classe aléatoirement. On envisage deux lois de probabilités : une loi uniforme et une loi normale propre à chaque dimension.

La figure 10 présente la précision des prédictions sur le jeu de test mesurée pour un modèle acoustique discret à une couche cachée de dimension 32 et deux modèles faisant des prédictions basées respectivement sur une loi uniforme et une loi normale, en fonction du nombre de catégories considéré pour effectuer la discrétisation. On remarque que la précision des prédictions diminue en fonction du nombre de classes discrétisant chaque dimension de l'espace acoustique. De manière générale, on peut remarquer que lorsque le nombre de classes augmente, la largeur de celles-ci diminue, augmentant ainsi la probabilité d'effectuer une erreur de classification, ce qui explique la décroissance de la précision. Le cas de la loi uniforme correspond à prendre une catégorie purement au hasard, tandis que celui de la loi normale correspond à privilégier celle contenant la moyenne. En comparaison, puisque le modèle acoustique discrétisé est toujours plus précis que les deux modèles aléatoires, on peut en déduire qu'il apprend une estimation de la distribution des données et que celle-ci est plus précise que l'approximation de chaque dimension par une loi normale. Il peut être

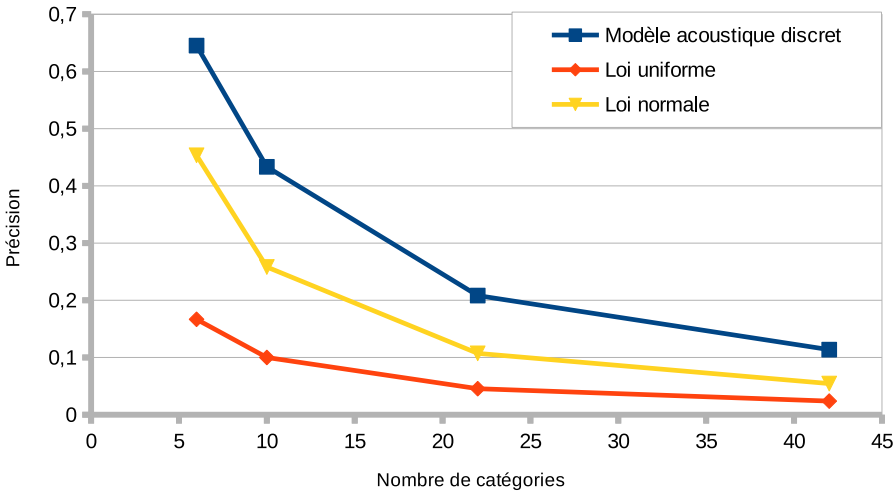

FIGURE 10 – Précision des prédictions d'un modèle acoustique discret en fonction du nombre de classes considérées, comparé avec les prédictions de modèles fondés sur une loi uniforme ou normale.

intéressant de remarquer que l'écart de précision entre le modèle acoustique discrétisé et le modèle aléatoire uniforme diminue en fonction du nombre de classes.

Une deuxième méthode de discrétisation envisagée consiste à appliquer une méthode de K-moyenne ou de mélange de gaussienne à K composantes pour partitionner l'espace acoustique en K clusters. De manière similaire on peut alors entraîner un réseau pour prédire à quel cluster appartient chaque unité. On ne développe pas plus cette idée car elle donne des résultats similaires à ceux obtenus par discrétisation et que cette méthode pose des problèmes de passage à l'échelle si l'on souhaite travailler sur des données massives.

Au final, on a tenté de discrétiser l'espace acoustique pour transformer un problème de régression en un problème de classification en espérant améliorer la qualité de l'apprentissage. Bien qu'il soit impossible de comparer directement le modèle acoustique classique et le modèle discret, on dispose désormais d'une seconde manière d'extraire des *embeddings* à caractère acoustique, on pourra donc comparer les deux modèles en fonction de la qualité de leurs *embeddings*.

2.2.5 Modèle multi-tâches

Dans cette section, on tente de mettre en place un modèle multi-tâches et d'évaluer ses performances. L'utilisation d'un modèle multi-tâches est motivée par deux raisons différentes. Dans un premier temps, la littérature montre que l'utilisation d'un modèle multi-tâches conduit en général à un modèle plus précis (*cf.* section 1.2.3) et la section précédente montre que nos modèles acoustiques manquent de précision, on espère donc améliorer leur performance à l'aide de l'utilisation de réseaux multi-tâches. Dans un second temps, on espère pouvoir moduler nos *embeddings* en leur faisant apprendre une représentation pertinente par rapport aux différentes tâches du réseau.

Bien que de nombreuses tâches soient envisageables, on se contente ici d'examiner un réseau multi-tâches tentant de prédire d'une part le vecteur de description linguistique et d'autre part le vecteur de description acoustique. Le choix de cette combinaison de tâches a été fait afin d'observer l'impact d'une information redondante (descripteurs linguistiques en entrée mais aussi en sortie)

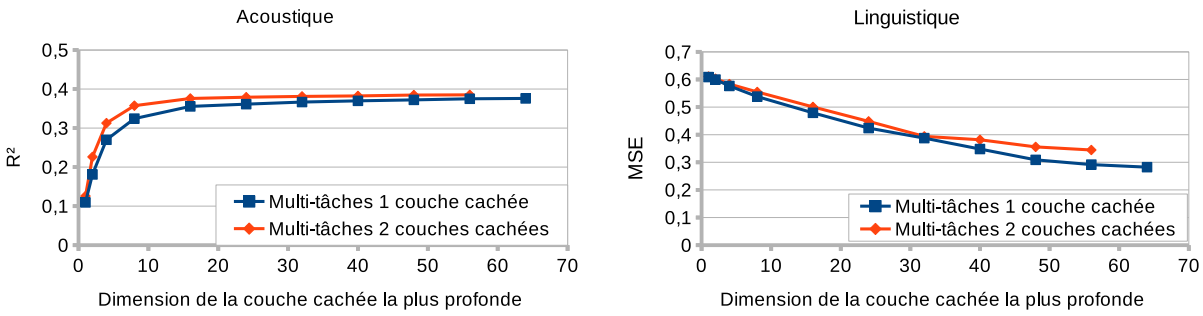

FIGURE 11 – Recherche de l'architecture optimale du réseau multi-tâches. À gauche l'évaluation de la tâche acoustique, à droite l'évaluation de la tâche linguistique.

sur la qualité des *embeddings* extraits. Afin d'obtenir l'architecture optimale de notre réseau multi-tâches, on entraîne des réseaux avec 1 ou 2 couches cachées, la sortie correspondant à deux couches linéaires indépendantes contenant respectivement 243 et 39 neurones. On entraîne nos réseaux pour minimiser la somme des MSE de la sortie linguistique et de la sortie acoustique, en considérant les deux tâches égales en termes de contribution : aucune n'est mise en avant par rapport à l'autre, elles ont le même poids dans la somme de la MSE.

La figure 11 présente à gauche les mesures de R^2 pour la tâche acoustique d'un modèle multi-tâches à une ou deux couches cachées, en fonction du nombre de neurones de la dernière couche cachée. On évalue la tâche acoustique avec le coefficient R^2 afin de pouvoir comparer les résultats à ceux du modèle acoustique par défaut. La figure présente à droite les mesures de MSE pour la tâche linguistique pour un modèle multi-tâches à une ou deux couches cachées en fonction du nombre de neurones dans la dernière couche cachée. En observant les résultats, l'architecture optimale en termes de nombre de couches cachée n'est pas évidente dans le cas du réseau multi-tâches. Dans le cas de la tâche acoustique, le modèle à 2 couches cachées est plus performant en basse dimension et équivalent en haute dimension au modèle à 1 couche cachée. Le phénomène opposé est observable dans le cas de la tâche linguistique : le modèle à 2 couches cachées est équivalent en basse dimension mais moins bon en haute dimension que le modèle à 1 couche cachée. On doit donc décider de l'architecture optimale en fonction de la tâche que l'on considère principale : celle à deux couches cachées dans le cas de la tâche acoustique, celle à une couche cachée dans le cas de la tâche linguistique. En revanche, notre but lors de ce stage est d'extraire des *embeddings* de basse dimension, ce qui pousse à privilégier l'architecture à deux couches cachées. Un phénomène intéressant peut être observé si l'on examine la MSE totale lors de l'entraînement des réseaux multi-tâches. La MSE due à la tâche linguistique est relativement faible, de l'ordre du dixième au centième, tandis que la MSE due à la tâche acoustique est élevée en comparaison, de l'ordre de la dizaine. Ainsi, bien que l'on n'ait pas mis de tâche en avant en formulant la fonction de coût générale du réseau comme la somme des MSE de chaque tâche, la différence d'ordre de grandeur dans les erreurs, mais aussi celui des données de base, poussent le réseau à favoriser la minimisation de la MSE associée à la tâche acoustique. Les effets de la modification de la pondération n'ont malheureusement pas été étudiés lors de ce stage.

On s'intéresse maintenant à vérifier si l'utilisation d'un modèle multi-tâches améliore vraiment la précision des prédictions dans notre cas par rapport à un modèle classique avec une seule tâche.

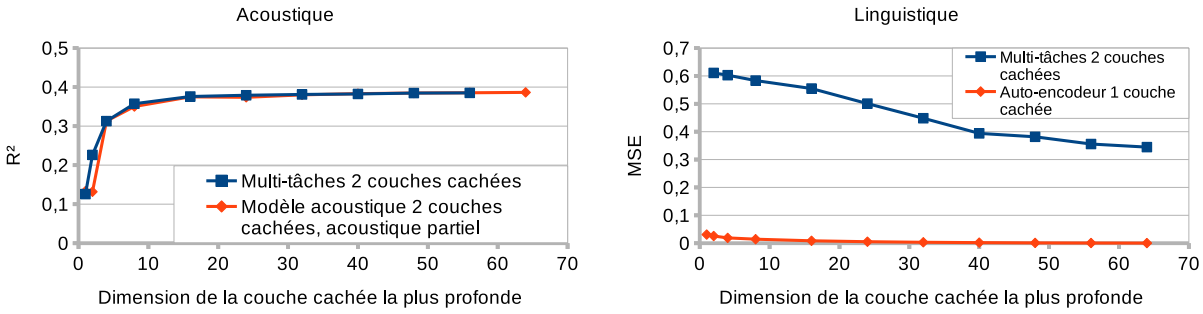

FIGURE 12 – Comparaison du réseau multi-tâches avec le modèle acoustique classique à gauche et avec l'auto-encodeur à droite.

Si l'on observe à gauche sur la figure 12 la comparaison entre les prédictions du multi-tâches et du modèle acoustique par défaut, on réalise que le réseau multi-tâches est au moins aussi précis que le modèle acoustique, l'amélioration visible n'étant pas significative. En revanche, si l'on observe à droite sur la figure 12 la comparaison entre les prédictions du multi-tâches et de l'auto-encodeur linguistique, on réalise que les performances du réseau multi-tâches sont très inférieures. Ceci peut être expliqué par la remarque faite précédemment : lors de l'apprentissage, le réseau a tendance à privilégier la diminution de la MSE de la tâche acoustique. Au final, au vu des résultats à l'issue de l'entraînement, on ne peut pas conclure que le réseau multi-tâches permet d'améliorer de manière significative les performances du modèle acoustique par défaut ou celle de l'auto-encodeur, mais il offre malgré tout une forme d'*embeddings* différents qui peuvent présenter un intérêt pour la synthèse de parole par sélection d'unités.

Au sein de cette partie, différents types de réseaux de neurones ont été entraînés dans le but d'en extraire des *embeddings*. Chaque type de réseau a été entraîné dans l'espoir d'apporter différentes bonnes propriétés à l'espace d'*embeddings* : l'auto-encodeur permet de définir une distance purement linguistique entre nos unités tandis que les différents modèles acoustiques (y compris le multi-tâches) permettent, on l'espère, de définir une distance aussi bien linguistique qu'acoustique entre nos unités. Cependant, nous n'avons pas encore évalué la qualité de nos *embeddings*, ni même défini la manière de les évaluer. C'est le sujet de la partie suivante.

3 Évaluation des embeddings

La question de l'évaluation des *embeddings* constitue un objectif majeur de ce stage. Bien que ceux-ci soient régulièrement utilisés dans de nombreuses publications, seuls leurs effets sont examinés. En revanche la manière d'obtenir un espace d'*embeddings* optimal ou même la manière d'évaluer un *embedding* de manière générale reste relativement peu, voire pas, documentée. En général, on considère avoir construit un bon *embedding* si la précision de la tâche originale est améliorée par l'utilisation des *embeddings*. Dans cette partie, on s'attache dans un premier temps à proposer une méthode d'évaluation générale et indépendante du problème en s'inspirant de la notion mathématique d'*embeddings*. On s'intéresse ensuite à proposer une méthode d'évaluation plus proche de notre problème mais applicable à d'autres : la conservation du rang de plus proches

voisins par projection dans un espace d'*embeddings*. Nous appliquons cette méthode d'évaluations sur les différents modèles étudiés en section 2 afin de conclure sur la viabilité de l'approche générale.

3.1 Évaluation de la distorsion spatiale

La notion d'*embeddings* est présente en mathématiques dans de nombreux domaines, en particulier en topologie différentielle mais aussi dans le domaine des espaces métriques. Les définitions varient entre domaines et au sein même d'un domaine mais, de manière générale, la notion d'*embeddings* correspond à trouver un *mapping* f entre un espace d'origine O et un espace projeté E. On s'intéresse ici au cas des espaces métriques et à la définition d'un bi-Lipschitz *embedding* [21] :

Définition. *Bi-Lipschitz embedding :*
Soient deux espaces métriques (O, d_O) et (E, d_E), un mapping $f : O \to E$ est un bi-Lipschitz embedding si pour un certain facteur d'étirement $C > 0$:

$$\forall p, q \in O,\ Cd_O(p,q) \leq d_E(f(p), f(q)) \leq DCd_O(p,q), w$$

où le paramètre $D \geq 1$ est appelé la distorsion de f.

En interprétant cette définition, un bi-Lipschitz *embedding* est un *mapping* entre un espace d'origine et un espace projeté pour lequel la distance entre deux points projetés correspond à la distance entre les points d'origines, ajustée par un coefficient d'étirement et en accordant une erreur maximale appelée la distorsion. Avec une telle définition des *embedding*, ceux-ci sont assimilés à une technique de réduction de la dimensionnalité et il paraît naturel de comparer des *embeddings* entre eux à partir de leur facteur d'étirement et surtout leur distorsion, mesure indiquant si un espace projeté conserve la structure de l'espace d'origine, ou tout du moins l'ordre de grandeur entre les distances. On peut par exemple trouver le résultat suivant [22] :

Théorème. *Théorème de Johnson et Lindenstrauss :*
Soit $X \subset \mathbb{R}^d$ un ensemble de taille n, $\epsilon \in]0, 1/2[$, alors il existe un mapping $f : X \to \mathbb{R}^m$ où $m = O(\epsilon^{-2} ln(n))$ tel que :

$$\forall x, y \in X,\ (1-\epsilon)||x-y||_2^2 \leq ||f(x) - f(y)||_2^2 \leq (1+\epsilon)||x-y||_2^2$$

En d'autres termes, f est un bi-Lipschitz embedding d'étirement $(1-\epsilon)$ et de distorsion $\dfrac{1+\epsilon}{1-\epsilon}$

La démonstration de ce théorème se base sur la projection aléatoire des points de l'espace d'origine, l'espace d'*embedding* associé est donc dépendant de l'espace et des points d'origine. Ainsi, ce théorème permet d'obtenir une majoration de la dimension nécessaire pour obtenir un espace d'*embedding* de distorsion souhaitée. Ce théorème est applicable dans notre cas car l'espace linguistique et l'espace acoustique jouant le rôle de X sont tous deux des sous-ensembles de $\mathbb{R}^d$ et on cherche à projeter les points de ces espaces dans un sous-ensemble de $\mathbb{R}^m$ avec m très inférieur à d. Cependant, le grand nombre d'unités à notre disposition ne joue pas en notre faveur. Si l'on souhaite obtenir un espace projeté de faible distorsion, on obtient une dimension associée importante. Par exemple, pour $\epsilon = 0.1$, on a $m = O(0.1^{-2} ln(390000)) = O(1287)$. On peut obtenir un espace projeté de dimension plus raisonnable si l'on accepte une distorsion plus importante. Pour $\epsilon = 0.49$,

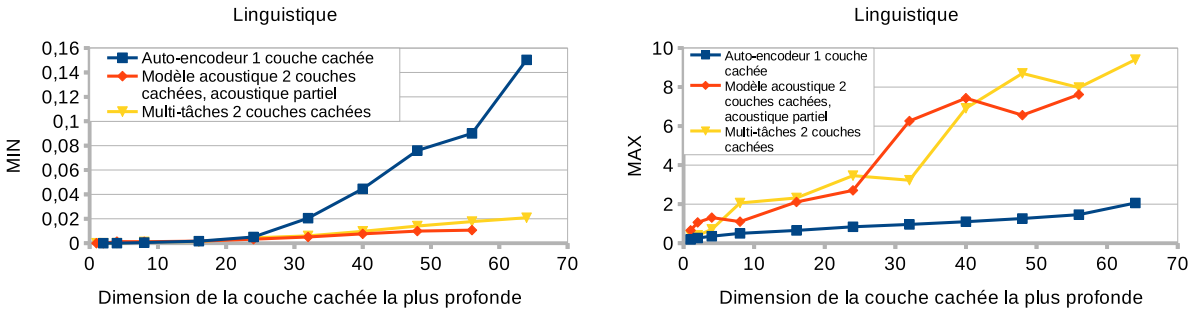

FIGURE 13 – Mesure de MIN et MAX pour les modèles de la section 2 en fonction de la dimension de la couchée la plus profonde, par rapport à l'espace linguistique. Plus MIN et MAX sont proches de 1 et plus le modèle est performant.

on a $m = O(0.49^{-2}ln(390000)) = O(54)$. Ce théorème met donc en avant la difficulté d'obtenir un espace projeté conservant les distances d'origines tout en effectuant une réduction de la dimensionnalité importante. Cependant, ce théorème étant basé sur la notion de projection aléatoire, il est intéressant de voir comment nos *embeddings* se comportent face au critère de distorsion à titre de comparaison.

Grâce à la définition d'un bi-Lipschitz *embedding* et au théorème de Johnson et Lindenstrauss, on dispose maintenant d'une méthode d'évaluation des *embeddings* entraînés dans la section 2 ainsi que d'une valeur de référence. Pour chacun des réseaux de neurones entraînés, on projette l'ensemble des points du jeu de test dans l'espace d'*embedding* défini par la couche cachée la plus profonde. On s'intéresse à mesurer le rapport entre les distances dans l'espace d'origine et la distance correspondante dans l'espace projeté. On s'intéresse en particulier aux valeurs suivantes :

$$MIN = \min_{p,q \in O}(\frac{d_E(f(p,f(q))}{d_O(p,q)}),$$

$$MAX = \max_{p,q \in O}(\frac{d_E(f(p,f(q))}{d_O(p,q)}).$$

On considère alors qu'un *embedding* est de qualité s'il conserve l'ordre de grandeur des distances par projection, ce qui se traduit par des valeurs de MIN et MAX proches de 1. On observe l'évolution de ces valeurs en fonction de la dimension de l'espace d'*embeddings* défini par chaque modèle entraîné dans la section 2.

La figure 13 présente l'évolution de MIN à gauche et MAX à droite sur le jeu de test en fonction du nombre de neurones de la couche la plus profonde pour la projection depuis l'espace linguistique vers l'espace d'*embeddings* défini par un auto-encodeur, un modèle acoustique et un réseau multi-tâches. Pour ces trois modèles, MIN augmente en fonction de la dimension de l'espace projeté, sans jamais atteindre 1. Cela signifie qu'au moins une partie des distances sont diminuées par projection dans l'espace d'*embeddings* mais que cette diminution s'atténue en fonction de la dimension. Puisque l'on cherche à avoir des valeurs de MIN proches de 1, on peut considérer que le modèle acoustique et le modèle multi-tâches ont des performances similaires, inférieures à celle de l'auto-encodeur. De même les valeurs de MAX augmentent en fonction de la dimension de l'espace de projection. Lorsque MAX est inférieur à 1, cela signifie que l'ensemble des distances

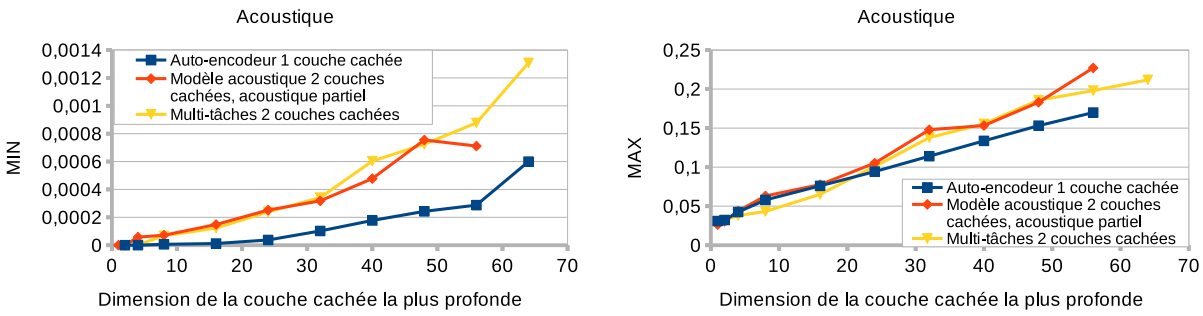

FIGURE 14 – Mesure de MIN et MAX pour les modèles de la section 2 en fonction de la dimension de la couchée la plus profonde, par rapport à l'espace acoustique. Plus MIN et MAX sont proches de 1 et plus le modèle est performant.

sont diminuées par projection tandis qu'une valeur supérieure à 1 signifie qu'au moins une partie des distances sont agrandies par projection dans l'espace d'*embeddings*. Puisque l'on cherche à avoir des valeurs de MAX proches de 1, là encore, l'auto-encodeur semble plus performant que le modèle acoustique et le réseau multi-tâches. Ce résultat est prévisible puisque l'auto-encodeur n'a été entraîné que sur les données linguistiques, il paraît logique que celui-ci soit plus à même de conserver les distances dans l'espace linguistique par projection dans l'espace d'*embeddings*. À titre de comparaison, selon le théorème de Johnson et Lindenstrauss, il est possible de trouver un *embedding* de dimension $O(54)$ tel que $MIN = 0,51$ et $MAX = 1,49$. Pour notre auto-encodeur de dimension 56, $MIN = 0,16$ et $MAX = 1,46$. Ainsi, la distorsion de notre auto-encodeur lorsque l'espace d'origine est l'espace linguistique est un peu moins bonne que la valeur trouvée théoriquement.

Dans l'expérience précédente, on a évalué la capacité de nos espace d'*embeddings* à conserver les distances de l'espace linguistique. On effectue ici un travail similaire : on reprend les projections des unités dans l'espace d'*embeddings* obtenues précédemment et on mesure les distances dans l'espace acoustique pour mesurer MIN et MAX en considérant l'espace d'origine comme l'espace acoustique. On s'attend cette fois à ce que les positions soient inversées et que le modèle acoustique et le modèle multi-tâches soient plus performants que l'auto-encodeur linguistique qui a été entraîné sans aucun contact avec les descripteurs acoustiques. Cependant, l'entraînement a montré que les modèles acoustiques et le réseau multi-tâches n'effectuent pas de prédictions précises et il ne serait donc pas étonnant que les *embeddings* issus de ces modèles ne conservent pas mieux les distances acoustiques que l'auto-encodeur.

Les résultats obtenus lors de l'expérience sont visibles sur la figure 14. On peut commencer par remarquer que les valeurs de MIN et MAX pour tous les modèles sont toujours inférieures à 1 ce qui signifie que toutes les distances de l'espace acoustique sont réduites par projection dans l'espace d'*embeddings*. De plus, les valeurs de MIN sont extrêmement faible par rapport à la valeur optimale de 1, ce qui pousse à se demander si désigner un modèle comme plus performant qu'un autre a réellement un sens ici : les distances ne sont clairement pas conservées lors de la projection.

Enfin, on s'intéresse aux modèles acoustiques discrets. Ceux-ci ont été entraînés comme une alternative au modèle acoustique classique dans l'espoir d'améliorer la précision de l'apprentissage

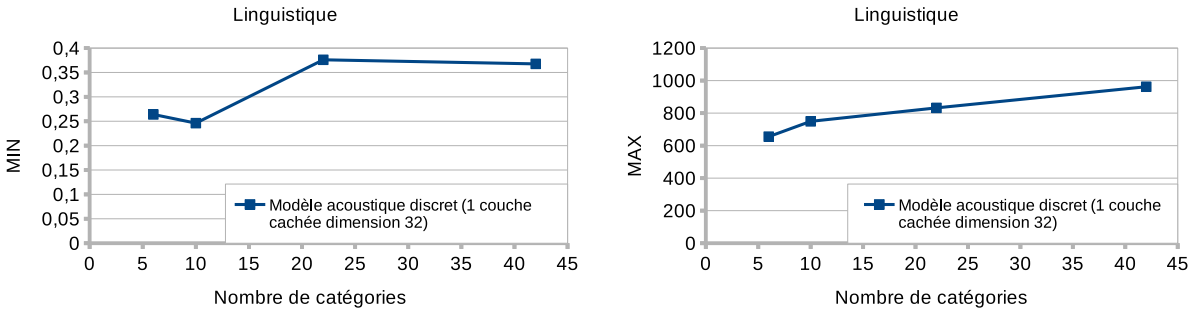

FIGURE 15 – Mesure de MIN et MAX pour le modèle acoustique discrétisé en fonction du nombre de catégories, par rapport à l'espace linguistique. Plus MIN et MAX sont proches de 1 et plus le modèle est performant.

et d'obtenir de meilleurs *embeddings*. On mesure donc à nouveau MIN et MAX dans l'espace linguistique. En observant les résultats sur la figure 15, on réalise rapidement que peu importe le nombre de catégories considérées, le modèle acoustique discret ne produit pas de meilleurs *embeddings* que le modèle classique. En effet, les distances dans l'espace projeté peuvent prendre 800 fois la valeur de la distance dans l'espace linguistique. Là ou le modèle acoustique par défaut a tendance à réduire les distances, le modèle discret à tendance à les agrandir. Bien que non montrés ici, les résultats sont similaires dans le cas des distances acoustiques.

Si au final les résultats obtenus dans cette section sont peu encourageants, cela provient en partie de la méthode d'évaluation envisagée. Il s'agissait ici de juger de la capacité d'un espace d'*embedding* à conserver l'ordre de grandeur des distances dans l'espace d'origine malgré une projection dans un espace de dimension plus réduite. En réalité, par rapport à la problématique de sélection d'unités, il n'est pas nécessaire que l'ensemble des distances soient conservées, on souhaite que les faibles distances restent faibles et il serait non seulement acceptable mais même préférable que les grandes distances soient allongées, le tout afin de faciliter la recherche de plus proches voisins. On se concentre donc dans la section suivante à mettre au point une méthode d'évaluation jugeant de la capacité d'un *embedding* à conserver les propriétés de plus proches voisins.

3.2 Conservation des plus proches voisins

On cherche ici à mesurer la capacité d'un *embedding* à conserver les plus proches voisins lors de la projection dans l'espace d'*embeddings*. L'idée générale des mesures qui sont proposées ici consiste à choisir au hasard un point du jeu de données de test et de classer la totalité des points du jeu d'entraînement (jouant ici le rôle de la base de données utilisée lors de la synthèse de parole) en fonction de leur distance par rapport au point choisi dans l'espace linguistique. On peut alors projeter le point choisi dans l'espace d'*embedding* et effectuer le classement correspondant dans l'espace projeté. Dans le cas idéal, le classement dans l'espace linguistique et dans l'espace projeté est le même, on peut donc mesurer la capacité d'un espace d'*embedding* à conserver le rang à l'issue d'une recherche de plus proches voisins en quantifiant la différence entre ces deux classements. Il serait intéressant ici d'utiliser une mesure de corrélation du rang, comme le τ de Kendall ou le ρ de Spearman [23]. Malheureusement, cette branche de la littérature a été découverte trop tard durant le stage pour être développée dans ce rapport, on présente ici les réflexions précédant cette

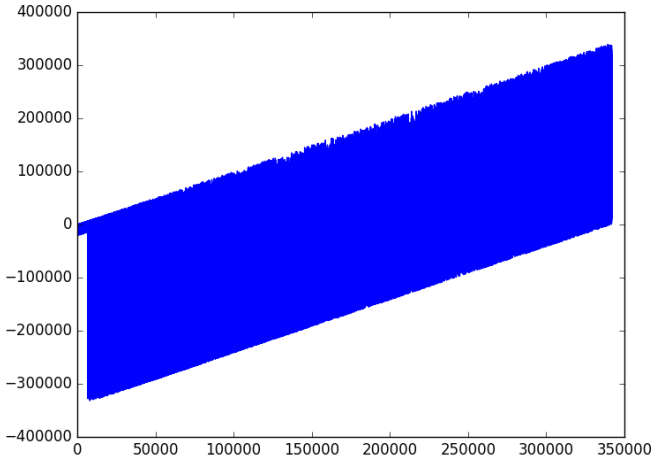

FIGURE 16 – Erreur sur le rang dans l'espace projeté en fonction du rang dans l'espace linguistique pour l'auto-encodeur linguistique de dimension 32.

découverte.

Dans la suite, on note o_i le rang du point i dans le classement issu d'une recherche de plus proches voisins dans l'espace d'origine et e_i le rang du point i dans le classement associé à l'espace projeté. On peut adopter deux points de vue différents quant à la conservation du rang à l'issue d'une recherche de plus proches voisins :

- Conservation exacte : $o_i = a \Rightarrow e_i = a$, un point doit avoir le même rang dans l'espace projeté que dans l'espace d'origine.
- Conservation relative : $o_i < a \Rightarrow e_i < a$, un point doit conserver sa position relative dans le classement par projection.

Il s'agit maintenant de définir une mesure d'erreur associée aux deux points de vue précédents et de les appliquer pour évaluer les différents espaces d'*embedding* à notre disposition.

3.2.1 Conservation exacte

Dans le cas de la conservation exacte de rang, il paraît normal de définir l'erreur comme $\delta_i = o_i - e_i$. On peut alors observer l'erreur commise pour chaque point du jeu d'entraînement en fonction de son rang dans le classement d'origine, pour des *embeddings* extraits du modèle auto-encodeur linguistique de dimension 32. On espère voir des erreurs faibles pour les premiers points du classement d'origine et une erreur grandissante en valeur absolue en fonction du rang dans le classement d'origine.

La figure 16 présente l'évolution de l'erreur δ_i en fonction de i le classement à l'issue d'une recherche des plus proches voisins d'un point aléatoire du jeu de test dans le jeu d'entraînement. La projection est effectuée dans l'espace d'*embeddings* induit par le modèle auto-encodeur de dimension 32. L'évolution de l'erreur s'effectue en deux temps : les points de rangs inférieur à 10 000 et ceux de rang supérieur. Commençons par ceux de rang supérieur à 10 000. Pour chaque point, l'erreur oscille entre une valeur haute et une valeur basse qui varient linéairement en fonction du rang dans le classement d'origine pour donner la forme trapézoïdale de la figure 16. Dans un premier

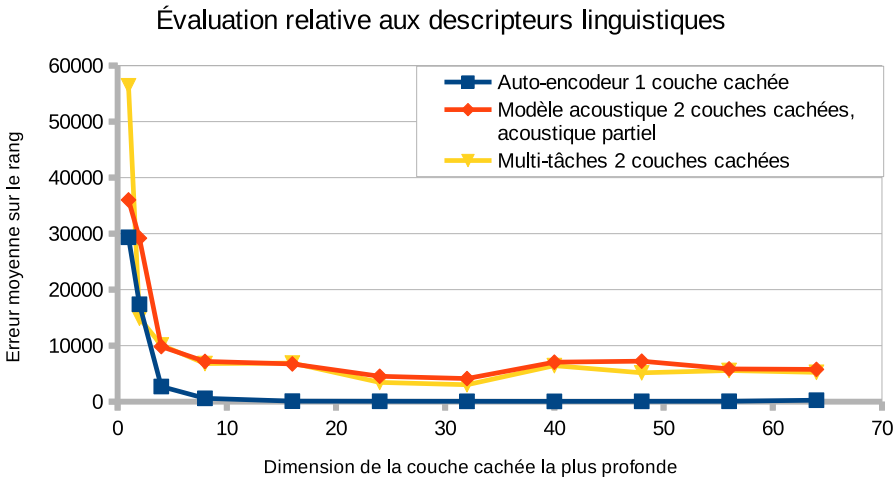

FIGURE 17 – Erreur moyenne sur le rang en fonction de la dimension pour différents modèles

temps, l'erreur à tendance à être négative ce qui est logique puisque dans le cas des points de rang faible ; il y a plus de chances d'obtenir un rang erroné supérieur qu'un rang erroné encore plus faible. Inversement, pour les points de haut rang, l'erreur a tendance à être positive pour la raison opposée. Si ces résultats semblent peu encourageants, il est intéressant de remarquer un phénomène différent pour les unités de très bas rang, inférieur à 10 000. Pour ces points, l'erreur est beaucoup plus proche de 0, limitée à 20 000 en valeur absolue. L'erreur paraît toujours importante mais doit être comparée au nombre moyen d'unités par classe de phonème, 10 000. Il paraît donc évident que le rang à l'issu d'une recherche de plus proches voisins n'est pas conservé par cet *embedding* pour des rangs suffisamment grand. Il semble cependant possible que nos *embeddings* conservent le rang des points en tête de classement, ce qui serait suffisant dans notre cas d'application. On se concentre donc désormais sur l'évaluation de la conservation des plus proches voisins des points en tête de classement.

On effectue la recherche des 100 plus proches voisins pour 100 points aléatoires du jeu de test et on s'intéresse à l'erreur moyenne sur le rang en fonction de la dimension de l'*embedding*. Si l'on considère qu'un espace projeté de qualité conserve totalement le rang des plus proches voisins, le meilleur modèle sera celui pour laquelle l'erreur moyenne est la plus proche de 0. Comme dans la section 3.1, on s'attend à ce que l'auto-encodeur linguistique soit le plus performant car on évalue ici la conservation des voisins linguistiques.

En observant les résultats sur la figure 17, on s'aperçoit que comme d'habitude, le modèle acoustique classique et le multi-tâches ont un comportement similaire et sont peu distinguables l'un de l'autre. Si l'on regarde les valeurs d'erreurs moyennes, celles-ci décroissent en fonction de la dimension de l'*embedding* pour atteindre un ordre de grandeur de 10 000 à partir de la dimension 4. Cette valeur est très élevée, suffisamment pour considérer que les *embeddings* associés ne conservent pas les 100 plus proches voisins lors d'une projection depuis l'espace linguistique. Si l'on se concentre sur l'auto-encodeur linguistique cette fois, on remarque à nouveau une décroissance, mais des valeurs bien plus basses : l'erreur moyenne est inférieure à 100 dès la dimension 16. Cela signifie qu'à partir de la dimension 16, si l'on effectue une recherche des 200 plus proches voisins

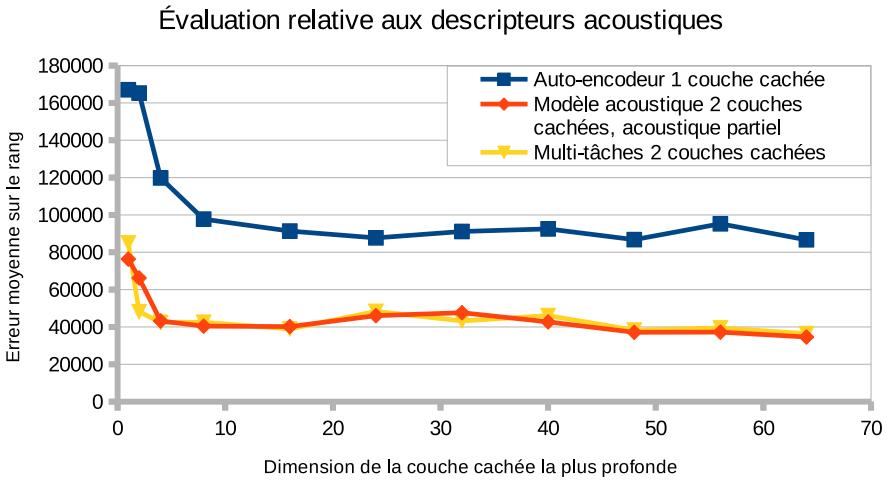

FIGURE 18 – Erreur moyenne sur le rang en fonction de la dimension pour différents modèles

dans l'espace d'*embedding*, il y a une forte probabilité que les 100 plus proches voisins de l'espace linguistique soient présents dans ces 200 points. On peut donc considérer que l'auto-encodeur linguistique conserve les 100 plus proches voisins bien que de manière imprécise, l'imprécision n'étant pas un problème si l'on met en place une reconnaissance des faux positifs.

Encore une fois, le modèle acoustique classique et le réseau multi-tâches se sont avérés décevant, observons si leur performance s'améliore lorsque l'on compare de manière similaire, le classement dans l'espace acoustique et le classement dans l'espace projeté. On s'attend à ce qu'ils conservent mieux la propriété de plus proches voisins acoustiques de l'auto-encodeur. Les résultats visibles sur la figure 18 montrent que dans le cas de l'auto-encodeur comme dans le cas du modèle acoustique classique et du multi-tâches, l'erreur moyenne diminue en fonction de la dimension de l'*embedding* vers une valeur limite, 100 000 et 40 000 respectivement. On pourrait donc dire que les deux modèles ayant été entraînés avec des données acoustiques conservent mieux les 100 plus proches voisins acoustiques, mais en réalité aucun des modèles ne conserve réellement cette propriété. On ne développera pas l'expérience ici mais il s'avère que le modèle acoustique discret conserve un peu mieux les distances acoustiques que le modèle classique. Il est intéressant de remarquer que le modèle discret retenu ne contient qu'une seule couche cachée tandis que le modèle classique en contient deux, une expérience intéressante consisterait à étudier l'importance du nombre de couches cachées sur la qualité d'un *embedding*.

3.2.2 Conservation relative

Dans cette section, on s'attache à mettre en place une mesure de la conservation du rang à l'issue d'une recherche de plus proches voisins en considérant qu'il y a conservation si $o_i < a \Rightarrow e_i < a$. Si on note $E_{o_i,a}$, respectivement $E_{e_i,a}$, l'ensemble des points ayant un rang inférieur à a dans l'espace d'origine, respectivement l'espace projeté, on peut ramener la mesure de véracité de l'implication

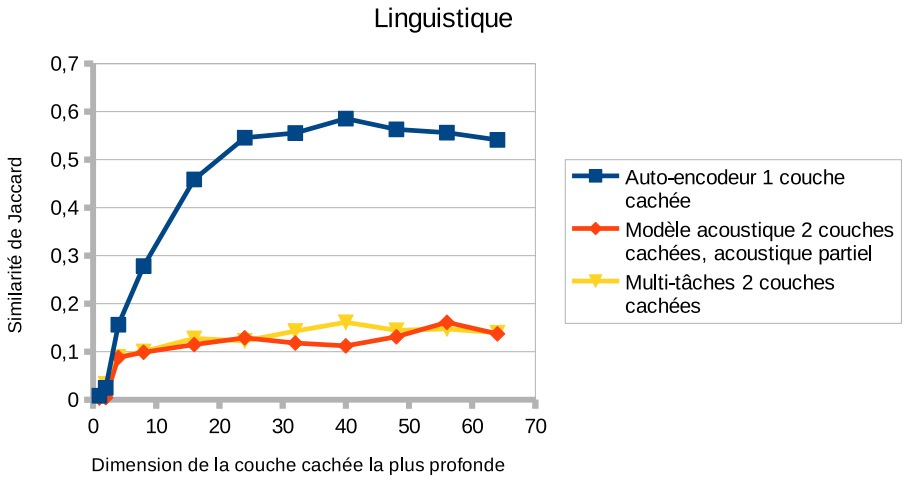

FIGURE 19 – Similarité de Jaccard moyenne en fonction de la dimension pour différents modèles

à la mesure de similarité entre les deux ensembles. On utilise ici la similarité de Jaccard :

$$J(E_{o_i,a}, E_{e_i,a}) = \frac{E_{o_i,a} \cap E_{e_i,a}}{E_{o_i,a} \cup E_{e_i,a}}.$$

Comme toute similarité, une valeur de 1 indique que les deux ensembles sont égaux et une valeur de 0 indique qu'ils sont complètement dissimilaires. Puisque l'on cherche à obtenir des *embeddings* qui conservent les plus proches voisins par projection, on cherche à maximiser la similitude.

Pour notre expérience, on considère que $a = 200$ afin de vérifier l'assertion faite dans la section 3.2.1 que dans le cas de l'auto-encodeur les 100 plus proches voisins dans l'espace linguistique sont dans les 200 plus proches voisins dans l'espace projeté pour l'auto-encodeur linguistique de dimension 32 et plus. On mesure la moyenne de la similarité de Jaccard sur 100 points pour nos différents modèles. Si l'assertion est vraie, on doit obtenir pour l'auto-encodeur un maximum autour de 0,5 à partir de la dimension 32.

L'observation des résultats sur la figure 19 montre qu'on obtient les résultats attendus et cohérents avec ceux de la mesure de conservation exacte de rang :

- Pour tous les modèles, la similarité augmente en fonction de la dimension de l'*embedding* pour tendre vers une valeur limite.
- Le modèle acoustique classique et le multi-tâches ont des performances similaires : seul 10% des 200 plus proches voisins linguistiques sont conservés par projection.
- L'auto-encodeur linguistique conserve mieux les plus proches voisins en atteignant 55% dès la dimension 24, ce qui est en accord avec l'affirmation de la section 3.2.1.

La même expérience dans l'espace acoustique donne des résultats en accord avec ceux obtenus jusqu'à présent : au mieux 1% des 200 plus proches voisins acoustiques sont conservés par projection.

Au final, on a mis en place deux mesures complémentaires de conservation de plus proches voisins. La première, basée sur une conservation exacte des plus proches voisins, donne une mesure de la dispersion des plus proches voisins par rapport à leur rang d'origine tandis que la deuxième, basée sur une conservation relative, donne une mesure du degré de conservation des plus proches

voisins au sein d'un intervalle. Il serait d'ailleurs intéressant d'observer le comportement de notre mesure et de nos modèles en fonction de la taille de l'intervalle choisi. Une fois ces mesures appliquées aux réseaux de neurones entraînés, il s'avère que les modèles acoustiques et le multi-tâches ne conservent pas les plus proches voisins, ce qui est probablement dû à leur manque de précision initial. En revanche, les résultats sur l'auto-encodeur sont encourageant : il conserve de manière imprécise les plus proches voisins. Il serait intéressant d'observer si les faux positifs sont des résultats réellement éloignés où s'ils constituent des candidats acceptables.

Conclusion

Le but de ce stage était d'effectuer un travail préliminaire à la conception d'un moteur de synthèse par sélection d'unités guidée par réseaux de neurones. Il s'agissait d'évaluer la capacité des *embeddings* issus de réseaux de neurones à définir un espace vectoriel où la distance euclidienne naturelle serait utilisée comme coût de sélection. Pour cette évaluation, deux critères ont été proposés : la conservation des distances dans l'espace linguistique et acoustique, ainsi que la conservation du rang à l'issue d'une recherche de plus proches voisins dans l'espace linguistique et l'espace acoustique. Si le premier critère paraît trop général, le second donne des résultats encourageants. En effet, bien que les modèles acoustiques entraînés ne semblent pas conserver les plus proches voisins linguistiques, l'auto-encodeur linguistique semble les conserver bien que de manière imprécise. Malheureusement, aucun des modèles entraînés ne conserve les plus proches voisins acoustiques, pas même les modèles acoustiques. Cependant, puisque les modèles acoustiques semblent mal entraînés, on ne peut pas conclure sur l'incapacité totale d'un espace d'*embedding* à conserver les plus proches voisins acoustiques.

Certaines expériences ont été laissées de côté par manque de temps, et pourront constituer le début de mon travail de thèse sur un sujet similaire. Pour commencer, afin de s'assurer que les modèles acoustiques sont entraînés correctement, il serait intéressant de reprendre un jeu de données utilisé dans la littérature et une architecture de réseaux déjà documentée. Alors, on pourrait déterminer si nos réseaux sont mal entraînés, si l'on manque de données, ou si la méthode pour inclure un aspect acoustique à nos *embeddings* est erronée. Une hypothèse implicite effectuée lors de ce stage est que la qualité d'un *embedding* est corrélée à la précision du réseau dont on l'a extrait, il serait intéressant de remettre cette hypothèse en question et d'évaluer les *embeddings* de réseaux qui ont été délaissés après l'entraînement. La parole est un flux continu et les unités de chaque énoncés ont ici été considérées indépendamment. On peut penser que prendre en compte l'aspect séquentiel de la parole utilisant des couches LSTM [24] pourrait améliorer nos résultats. Enfin, aucune synthèse de parole n'a été effectuée en utilisant les coûts de sélection d'*embeddings*, il aurait été intéressant d'évaluer nos *embeddings* en fonction de la parole synthétisée en supposant que ceux-ci sont déjà suffisamment bon pour obtenir une parole intelligible.

Bien que les résultats à l'issue de ce stage soient mitigés, l'utilisation d'*embeddings* pour définir un coût de sélection automatiquement semble être un sujet de recherche très actuel comme le montre les travaux de [25] qui seront publiés lors de la prochaine conférence Interspeech. En fonction de son contenu, ce papier pourrait s'avérer un bon point de départ et de comparaison pour le travail de thèse.

Références

[1] Jacob Benesty, M Mohan Sondhi, and Yiteng Huang. *Springer handbook of speech processing*. Springer Science & Business Media, 2007.

[2] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6) :453–467, 1990.

[3] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech and Signal Processing (ICASSP), 1996 IEEE International Conference on*, volume 1, pages 373–376. IEEE, 1996.

[4] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11) :1039–1064, 2009.

[5] Yao Qian, Yuchen Fan, Wenping Hu, and Frank K Soong. On the training aspects of deep neural network (dnn) for parametric tts synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3829–3833. IEEE, 2014.

[6] Keiichi Tokuday and Heiga Zen. Directly modeling speech waveforms by neural networks for statistical parametric speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4215–4219. IEEE, 2015.

[7] Stas Tiomkin, David Malah, Slava Shechtman, and Zvi Kons. A hybrid text-to-speech system that combines concatenative and statistical synthesis units. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, volume 19, pages 1278–1288. IEEE, 2011.

[8] Thomas Merritt, Robert AJ Clark, Zhizheng Wu, Junichi Yamagishi, and Simon King. Deep neural network-guided unit selection synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5145–5149. IEEE, 2016.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[11] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011.

[12] Leonardo Badino. Phonetic context embeddings for dnn-hmm phone recognition. In *INTERSPEECH*, pages 405–409, 2016.

[13] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4460–4464. IEEE, 2015.

[14] Michael L Seltzer and Jasha Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6965–6969. IEEE, 2013.

[15] Jon Louis Bentley and Jerome H Friedman. Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4) :397–409, 1979.

[16] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science (FOCS), 2006 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[17] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. In *IEEE Transactions on Pattern Analysis and Machine Intelligence on*, volume 36, pages 2227–2240. IEEE, 2014.

[18] Md Afzal Hossan, Sheeraz Memon, and Mark A Gregory. A novel approach for mfcc feature extraction. In *Signal Processing and Communication Systems (ICSPCS), 2010 4th International Conference on*, pages 1–5. IEEE, 2010.

[19] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude. *COURSERA : Neural networks for machine learning*, 4(2) :26–31, 2012.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow : Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv :1603.04467*, 2016.

[21] Piotr Indyk and Assaf Naor. Nearest-neighbor-preserving embeddings. *ACM Transactions on Algorithms (TALG)*, 3(3) :31, 2007.

[22] Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. *CoRR*, abs/1609.02094, 2016.

[23] Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th international conference on World wide web*, pages 571–580. ACM, 2010.

[24] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget : Continual prediction with lstm. 1999.

[25] Vincent Wan, Yannis Agiomyrgiannakis, Hanna Silen, and Jakub Vit. Google's next-generation real-time unit-selection synthesizer using sequence-to-sequence lstm-based autoencoders. In *Interspeech*, 2017.

A Annexes

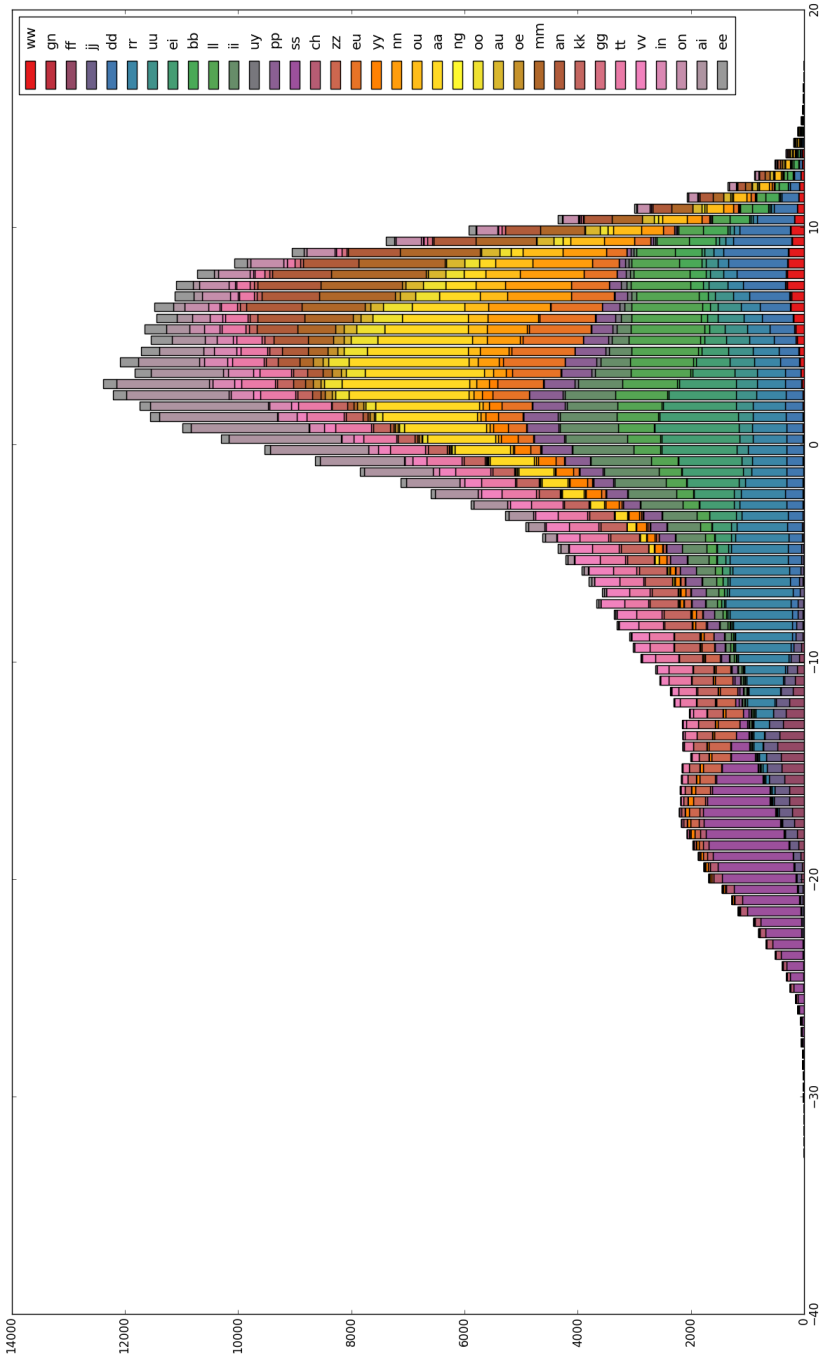

FIGURE 20 – Histogramme du premier coefficient acoustique statique. La proportion de chaque classe de phonème au sein de chaque catégorie est indiquée par une couleur différente.